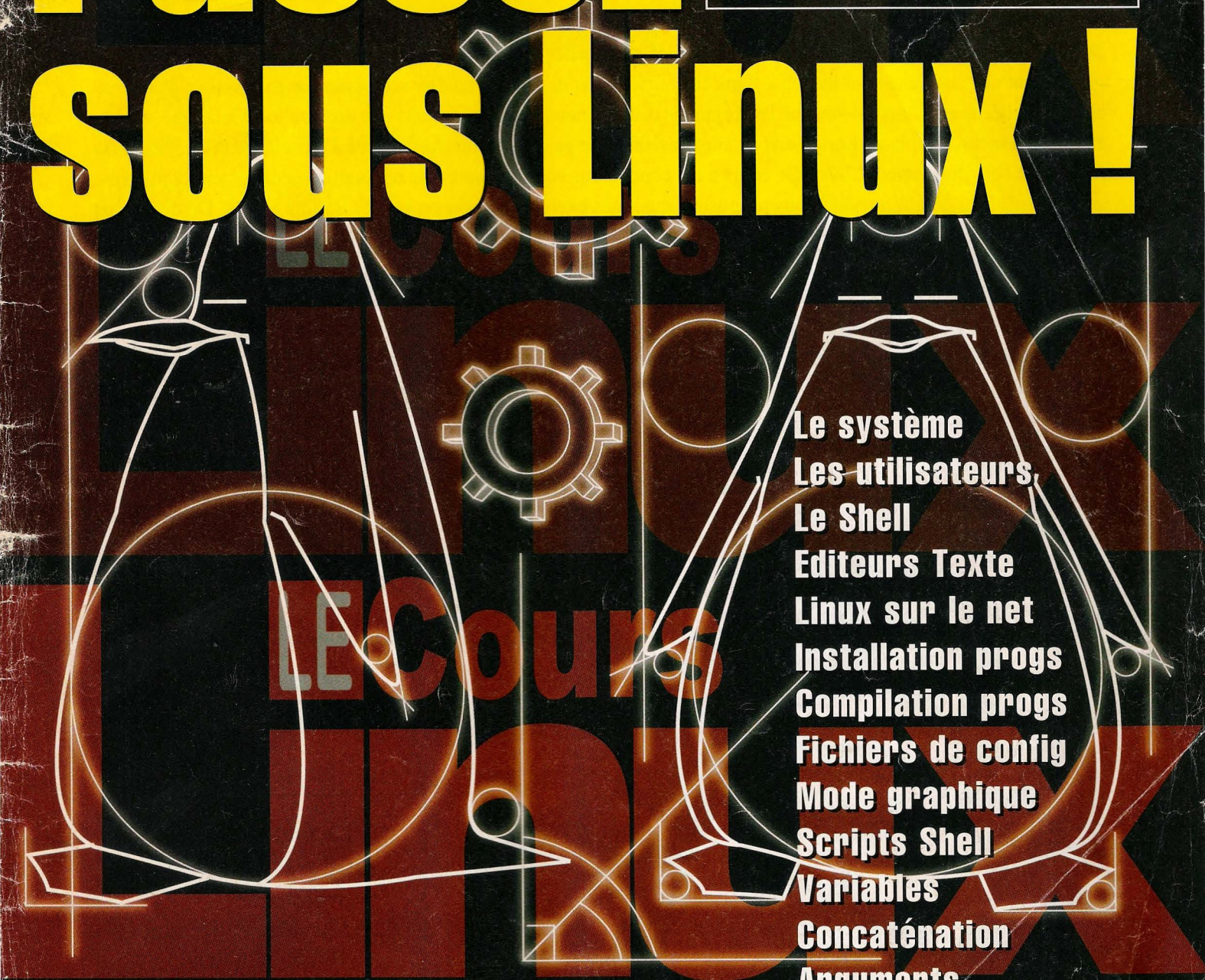
THE HACKADEMY
PROG
Juillet - août - n°3 / 6 euros

Exclusif :
Le cours officiel de The Hackademy School

Passez sous Linux !



- Le système
- Les utilisateurs
- Le Shell
- Editeurs Texte
- Linux sur le net
- Installation progs
- Compilation progs
- Fichiers de config
- Mode graphique
- Scripts Shell
- Variables
- Concaténation
- Arguments
- Instruction read
- Les Itérations
- Exercices

LE Cours Linux

Level .1

Passez sous Linux avec le cours officiel de The Hackademy !

Pour la première fois, nous diffusons auprès du grand public le cours Linux dispensé à the Hackademy School. Il s'agit ici de la première partie, dite " Niveau 1 ".
Son objectif est de vous permettre de découvrir les principales commandes et utilisations de Linux. Pour profiter pleinement de l'enseignement dispensé dans ce cours, que vous pouvez étudier seul, vous devez disposer d'une machine installée avec Linux. Si vous ne l'avez jamais installé, nous vous conseillons une distribution Mandrake, simple et rapide à mettre en place. Lorsque vous aurez assimilé ce premier cours, vous serez à l'aise avec toutes les commandes et l'environnement Linux. Ce cours a été conçu pour vous permettre de débuter immédiatement et progressivement. En fin de cours, nous vous proposons quelques exercices pratiques sur les Scripts Shells. De quoi vous entraîner cet été avant de passer, si le cœur vous en dit, à la partie dite de " Niveau 2 " qui est actuellement dispensée à The Hackademy School Paris et Marseille. Elle également accessible en cours par correspondance. Pour toute information sur ces formations, vous pouvez consultez le site www.thehackademy.net, nous contacter au :
01 40 21 01 20, ou nous envoyer un mail (hackademy@thehackademy.net).
La communauté de The Hackademy peut également répondre à vos questions sur le forum très actif disponible sur le site.

Bon cours à tous, à bientôt dans la communauté Linux, et bonnes vacances studieuses !

The Hackademy Team

Le cours Linux de the Hackademy
est édité par D.M.P
26 bis rue Jeanne d'Arc
94160 St Mandé

Rédaction :
Hackademy Team
Maquette :
Weel
Directeur de Publication :
O. Spinelli

Imprimé en CE

© DMP

Sommaire cours linux

PARTIE I : APPROCHE THÉORIQUE. P.4

I) Linux : définition

II) Le système de fichier sous Linux

III) Les utilisateurs

PARTIE II : PRATIQUE. P.12

I) Le Shell

II) Comprendre le shell

PARTIE III : UTILISATION QUOTIDIENNE. P.23

I) les éditeurs de texte

II) Linux sur le net

III) Introduction à l'installation et à la compilation d'un programme

IV) Les fichiers de configuration

V) Utilisation du mode graphique

PARTIE IV : LES SCRIPTS SHELL. P.41

1) Comment les définir ?

2) Les variables

3) La concaténation

4) Les arguments

5) L'instruction read

6) Les itérations

7) Exercices

PARTIE I : Approche théorique

I) Linux : définition

1) Historique de Linux

Linux a été inspiré de Minix, qui était en 1991 développé par Linus Torvalds un jeune étudiant âgé de 23 ans. Linus avait envie de développer un clone du système d'exploitation nommé Unix (système propriétaire), qui avait un large succès dans les universités américaines. Linux, contrairement à beaucoup de systèmes d'exploitation, est gratuit et constamment développé par une énorme communauté de passionnés d'informatique se situant un peu partout dans le monde depuis que Linus a distribué les sources sur le web sous la licence GPL du projet GNU. Elle permet à n'importe qui de modifier, améliorer et redistribuer le programme. La plupart des applications sous licence GPL ont été portées sur Linux. La licence GPL est consultable sur le web :
http://www.linux.org/info/gnu.html

2) Les distributions de Linux

Il existe de nos jours une multitude de distributions Linux destinées à différents publics. Une distribution contient le noyau Linux, une aide, un support et d'autres fonctions, propres à la distribution. Donc chaque distribution a ses avantages et inconvénients, on peut séparer ces distributions en plusieurs groupes suivant l'utilisation que l'on désire en faire.

Utilisation bureautique

La Red Hat et la Suse sont les distributions les plus utilisées par les personnes qui désirent avoir un Linux stable, fiable ayant une multitude de paquetages, prêt à l'emploi pour les tâches bureautiques. Redhat reste tout de même une distribution très répandue pour les serveurs d'entreprises.
http://www.redhat.com
http://www.suse.com

Utilisation débutante

Pour ceux qui désirent découvrir Linux sans avoir à taper une ligne de commande, la Mandrake et la Caldera sont là. La Mandrake est la distribution que l'on trouve souvent chez les particuliers. Installation simple tout en français, configuration automatique des périphériques, interface conviviale ressemblant étrangement à un autre OS de chez Microsoft.
http://www.mandrakelinux.com/fr/
http://www.caldera.com/

Utilisation orientée serveur

La Debian est une distribution sécurisée permettant d'installer/mettre à jour rapidement différents serveurs, immédiatement disponibles après une seule commande. En effet, la Debian dispose d'une commande permettant d'installer/réinstaller n'importe quel paquetage (plus de 8000 disponibles) à partir du Web (ou autre support physique) tout en respectant leurs dépendances. Cette distribution n'est en revanche pas aussi simple à installer/configurer que les précédentes distributions. Ce qui fait de la Debian une distribution réservée à ceux qui connaissent déjà Linux et qui n'ont pas peur de passer des soirées à configurer leur matériel.
http://www.debian.org

Utilisation personnalisée

La Slackware est livrée avec énormément de paquetages de bas niveau, ce qui permet de se faire un système taillé sur mesure, en toute simplicité avec Yast, l'outil majeur de cette distribution. La priorité essentielle de cette distribution est la stabilité.
http://www.slackware.com/

Toutes ces distributions sont téléchargeables gratuitement sur le web ou vendues en coffret de plusieurs cdroms par correspondance.

3) La notion de serveur et système UNIX

UNIX est né en 1969 grâce à Ken Thompson. Ecrit d'abord en langage machine puis en assembleur par John Kernighan, il fut ré-écrit en C par Dennis Ritchie. Unix est utilisé dans le monde entier pour ses capacités en tant que serveur. Il n'est tout de même pas accessible à tout le monde étant donné qu'il s'agit d'un système propriétaire. Un serveur permet à une machine de proposer des services à des clients se trouvant sur le réseau LAN de l'entreprise ou à n'importe qui se trouvant connecté à Internet suivant nos désirs. Pour que le serveur puisse recevoir des demandes de connexion, il faut qu'il ouvre un de ses ports. Je vous rappelle qu'il existe 65535 sur chaque machine connectée à Internet, plus précisément, utilisant TCP/IP. Pour plus de détails référez-vous au cours "Hack1" sur les protocoles de la suite TCP/IP.

II) Le système de fichier sous Linux

Linux c'est, à la base, un noyau qui doit être configuré de manière à prendre en charge les périphériques et donner à l'utilisateur un moyen de communication avec le système. Ce moyen de communication c'est le shell, c'est-à-dire un invite de commande. Pour fonctionner correctement, le noyau a aussi besoin d'un système de fichiers hiérarchique. Sous Linux tout est fichier. C'est-à-dire qu'un fichier peut aussi bien être un simple texte, qu'un fichier représentant un périphérique matériel (disque dur, carte graphique etc...).

1) Le système de partition

Comme tout système d'exploitation, Linux utilise son propre système de partitions. Il peut fonctionner sur une seule partition même s'il est recommandé de créer ce que l'on appelle une partition swap. Cette partition swap est considérée par Linux comme une extension de la mémoire vive. En général, on choisira une taille de swap égale à deux fois la taille de la mémoire vive. La partition de type ext2 sera utilisée comme partition racine du système. Linux, contrairement à Windows, n'utilise pas du tout de lettre pour désigner une partition.

Sous Windows, on peut avoir c : et d : comme partitions.
c : *désignera la première partition*
d : *la seconde.*

Sous Linux, on aura par exemple hda1 et hda2.
hda1 : *désignant la première partition*
hda2 : *la seconde du premier disque dur*

hdb désignera donc le second disque dur. hdb2 la seconde partition du second disque dur. Le premier lecteur de disquette sera nommé fd0 et le second fd1, etc...Mais ce n'est pas parce que Linux a besoin absolument de partition ext2 ou ext3 pour fonctionner qu'il ne reconnaît pas d'autres types de partitions. Avec Linux vous pourrez accéder par exemple à toutes vos partitions MS-DOS, FAT32, NTFS et bien d'autres...

2) L'arborescence des répertoires

Toutes les distributions de Linux n'ont pas exactement la même arborescence, mais on retrouvera toujours certains répertoires indispensables. Il faut savoir que sous Linux la base de système, que l'on appelle racine, se représente par un "/". La racine est le point de montage principal pour tout système Linux, c'est à partir de ce point que tous les autres fichiers vont venir se "monter". Voici donc l'arborescence des répertoires de base que l'on trouve sur la plupart des distributions : voir tableau page suivante.

Répertoire			Description
/			Racine du système. C'est à partir de cette racine que sont montés tous les autres répertoires.
	/bin		Répertoire contenant les divers exécutables système mis à disposition des utilisateurs normaux. N'importe quel utilisateur (par défaut) pourra donc exécuter tout ce qui sera contenu dans ce répertoire.
	/boot		On trouvera dans ce répertoire les éléments indispensables au démarrage de Linux, comme le noyau par exemple
	/cdrom		Répertoire utilisé comme point de montage pour le lecteur de cdrom
	/dev		Répertoire contenant des fichiers de périphériques (points d'accès aux divers périphériques du système). Pour accéder au lecteur de disquette par exemple, le noyau utilisera /dev/fd0.
	/etc		C'est ici que sont stockés les fichiers de configuration du système.
	/floppy		Répertoire utilisé comme point de montage pour le lecteur de disquette.
	/home		Répertoire contenant les répertoires personnels de chaque utilisateur.
	/lib		Répertoire des librairies communes utilisées par le système.
	/mnt		Répertoire utilisé pour monter les différents périphériques comme un disque amovible, une autre partition, cdrom, lecteur de disquette etc...
	/proc		Ici se trouvent les informations sur le fonctionnement de votre système. En réalité c'est un système de fichiers virtuel. On peut par exemple avoir des informations sur les périphériques pci via /proc/pci.
	/root		Répertoire personnel du root (administrateur/superutilisateur).
	/sbin		Répertoire contenant les différents exécutables réservés au superutilisateur.
	/tmp		C'est ce répertoire que le système utilisera pour stocker les fichiers temporaires.
	/usr		Répertoire contenant les programmes non système, librairies et autres fichiers accessibles à tous les utilisateurs.
		/X11R6	Répertoire contenant les exécutables, librairies, manpages du système d'affichage X Windows.
		/bin	On trouve dans ce répertoire tous les exécutables utilisateurs.
		/include	Répertoire contenant les fichiers d'en-tête utilisateurs, nécessaires à la recompilation d'un programme utilisateur.
		/lib	Répertoire contenant les librairies utilisées par les programmes utilisateurs.
		/local	Ici on trouvera les différents programmes, librairies, fichiers de configuration, etc... installés par les utilisateurs.
		/man	Répertoire où sont stockés les manpages. Les manpages sont en fait des fichiers d'aide permettant d'avoir la description complète d'une commande, librairies ou fichier de configuration.
		/sbin	Répertoire contenant les exécutables non-système, destinés principalement à être utilisés par le superutilisateur.
		/share	Répertoire contenant divers programmes partagés par les utilisateurs.
		/src	Ici seront stockés par le root les fichiers sources à compiler.
	/var		Répertoire contenant les fichiers qui ont tendance à changer de taille.
		/log	Contient divers fichiers log. Les fichiers log permettent de garder une trace de ce qui s'est passé sur le système. On y trouve, entre autres, auth.log (log des utilisateurs s'étant loggués sur le système), syslog (logs du système).
		/spool	Le répertoire spool a pour but de stocker temporairement les fichiers en attente comme les mails à envoyer ou les fichiers à imprimer.
		/mail	Contient les emails des utilisateurs
		/cron	**Contient les fichiers cron des utilisateurs**

3) Montage et démontage d'un système de fichier

Si l'on désire accéder à une partition autre que la partition racine du système, il va falloir la monter. C'est exactement la même chose pour le lecteur de cdrom, lecteur de disquette, etc.. Le terme "monter" signifie que l'on associe un périphérique, partition à un répertoire du système. Pour monter et démonter, on utilisera "mount" et "umount" comme dans l'exemple ci-dessous pour la partition hda4.

```
$ mount /dev/hda4 /mnt
```

On monte la partition hda4 dans le répertoire"/mnt". S'il vous indique qu'il ne reconnaît pas le format de votre support, vérifiez que votre noyau le prend bien en charge. Pour spécifier le format, utilisez l'option "-t" suivie du type, comme ceci pour une partition FAT32 :

```
$ mount /dev/hda4 /mnt -t vfat
```

Pour démonter une partition rien de plus simple :

```
$ umount /dev/hda4
```

Il est possible de demander à Linux de charger automatiquement une partition au démarrage grâce au fichier /etc/fstab dont voici le contenu pour ma machine :

```
$ more /etc/fstab

# /etc/fstab: static file system information.
#
# <file system> <mount point> <type> <options> <dump> <pass>
/dev/hda6 / ext2 defaults,errors=remount-ro 0 1
/dev/hda7 none swap sw 0 0
/dev/fd0 /floppy auto defaults,user,noauto 0 0
/dev/cdrom /cdrom iso9660 defaults,ro,user,noauto 0 0
/dev/hda1 /mnt/win ntfs default 0 0
```

Pour exemple ici, la partition hda6 de format "ext2" a comme point de montage "/".
hda7 est la partition de swap (c'est la seule partition qui n'a pas de point de montage).
fd0 le lecteur de disquette que l'on monte dans le répertoire /floppy.
cdrom le lecteur de cdrom que l'on monte dans le répertoire /cdrom.
hda1 la partition ntfs que l'on monte dans le répertoire "/mnt/win".

4) Le PATH

Il faut bien faire la différence entre un chemin relatif et un chemin absolu. Pour cela, nous allons prendre le cas où l'on désire lancer un exécutable nommé "aterm" et que l'on se situe dans le répertoire /usr. Nous allons voir un peu plus loin comment se présente le "shell", pour le moment retenez juste que le shell est représenté par le caractère "$", donc il n'est pas à taper dans tous les exemples qui vont suivre. Les lignes ne contenant pas le caractère "$" sont les résultats des commandes.

Le chemin absolu est le chemin complet en partant de la racine, plus le nom de l'exécutable:

```
$ /usr/bin/aterm
```

Le chemin relatif est le chemin complet de l'exécutable mais en partant du répertoire courant, ici on se trouve dans le répertoire /usr :

```
$ bin/aterm
```

Si l'on se trouvait dans un sous-répertoire de /usr/bin, prenons par exemple le répertoire"test/", on pourrait exécuter "aterm" en tapant :

```
$ ./../aterm
```

Le problème c'est qu'il faut tout de même taper à chaque fois le chemin complet, ce qui peut s'avérer très embêtant à la longue... C'est pour éviter cela que Linux utilise une variable d'environnement appelée PATH. Cette variable est définie pour tous les utilisateurs dans le fichier /etc/profile qui est uniquement modifiable par le root. Les fichiers .profile et .bash_profile pouvant se trouver dans le répertoire home (ex: /home/crashfr/) de chaque utilisateur, permettent aux utilisateurs de personnaliser leur profile. Si les 2 fichiers sont présents dans le répertoire home, .bash_profile est prioritaire, donc profile sera ignoré.

La variable PATH

La variable $PATH contient plusieurs chemins qui indiqueront au shell où il devra aller chercher les commandes à exécuter. Ce qui évite de taper le chemin complet à chaque fois que l'on désire lancer un programme. $PATH fait partie des variables d'environnement.

Commande	*Description*
echo $PATH	Affiche la variable $PATH
env	Affiche les variables d'environnement.

Exemple :

```
$ echo $PATH

/usr/local/bin:/usr/bin:/bin/usr/X11R6/bin
```

Donc dans le cas ci-dessus, quand l'utilisateur tapera une commande sans indiquer son chemin, Linux recherchera en premier dans le répertoire /usr/local/bin, ensuite dans /usr/bin et pour finir, dans /bin/usr/X11R6. Si jamais l'utilisateur veut rajouter un répertoire dans la variable PATH, il lui suffira de taper :

```
$ PATH=$PATH":/home/crashfr/bin"

$ echo $PATH

/usr/local/bin:/usr/bin:/bin/usr/X11R6/bin:/home/crashfr/bin
```

Exécuter un programme

Il existe trois manières différentes pour exécuter un programme sous Linux.
La première est en indiquant le chemin complet de l'exécutable :

```
$ /bin/ls
```

Va exécuter la commande "ls" se trouvant dans le répertoire /bin.

La deuxième nécessite un "./" devant le nom de notre fichier à exécuter.
Par exemple, si nous nous trouvons dans le répertoire /bin :

```
$./ls
```

Si je me trouve dans un répertoire où "ls" n'est pas présent, même si "ls" se trouve dans un des répertoires de la variable PATH, la commande ne sera pas exécutée. Donc, en mettant un "./" devant le nom de la commande, celle-ci sera exécutée à partir du répertoire courant sans utiliser la variable PATH.
La dernière manière est en tapant tout simplement le nom de la commande :

```
$ ls
```

Dans ce cas, la commande est recherchée dans un des répertoires définis dans la variable PATH mais pas dans le répertoire courant. Si la commande "ls" se trouve dans un des répertoires de la variable $PATH, "ls" sera exécuté à partir de n'importe quel répertoire du système.

III) Les utilisateurs

Linux est un système multi-utilisateurs et multitâches. Linux permet à ses clients (les utilisateurs) de travailler en même temps, de n'importe où, tout en ayant leurs propres répertoires et fichiers personnels. Multitâches, car un utilisateur peut très bien rédiger un roman et surfer sur le Web dans une autre fenêtre ou terminal. MSDOS est un système monotâche par exemple, il ne permet pas de lancer plusieurs applications en même temps. Chaque utilisateur sous Linux a des droits bien précis par rapport à chaque fichier présent sur le serveur. Un utilisateur ne peut pas modifier (par défaut) les fichiers appartenant à d'autres utilisateurs, le root peut en revanche autoriser, interdire à n'importe quel utilisateur de lire, écrire, exécuter un fichier.

1) L'identification des utilisateurs sous Linux

La notion de numéro d'identification
Sous Linux tous les utilisateurs, processus, etc... sont associés à des numéros. Les utilisateur, ont chacun un UID et un GID. L'UID est unique pour chaque utilisateur et permet à Linux de les différencier. Le GID identifie le groupe auquel appartient l'utilisateur. Suivant le groupe auquel il appartient, il aura plus au moins de droits sur l'ensemble du système.

-Le fichier passwd
Le fichier "/etc/passwd" contient plusieurs informations sur les utilisateurs pouvant se logguer sur le système. Chaque ligne du fichier représente un utilisateur. Par exemple pour l'utilisateur nommé "crashfr":

```
crashfr:x:1000:1000:HZV Team Member:/home/crashfr:/bin/sh
```

Format du fichier passwd :
`login:password:UID:GID:nomcomplet:repertoireperso:shell`

Commande	*Description*
cat /etc/passwd	affiche le contenu du fichier passwd

Sur les anciennes versions de Linux, le password des utilisateurs était conservé dans le fichier passwd de façon cryptée. Sur les versions récentes, on utilise ce que l'on appelle un fichier "shadow" pour les stocker. Si le fichier passwd contient un "x" à la place des passwords utilisateurs, c'est que les passwords sont stockés dans un fichier shadow. Par défaut le fichier passwd est accessible en lecture pour tout le monde, ce qui permet à n'importe quel utilisateur d'essayer de cracker le mot de passe du root ou d'autres utilisateurs. D'où la nécessité d'utiliser le fichier shadow qui n'est pas accessible en lecture, sauf pour le root.

Le fichier group
Le fichier /etc/group contient les différents groupes existant sur le système.

Exemple pour le groupe "sudo":
```
sudo:x:27:crashfr,xdream
```

Format du fichier group:
```
nomdugroupe:x:GID:membresdugroupe
```

Pour ajouter un nouveau groupe, il suffit de rajouter une ligne dans le fichier /etc/group en respectant le format ci-dessus sachant que chaque groupe a un GID unique. Pour rajouter un utilisateur dans un groupe, il y a deux façons. Soit en changeant le GID de l'utilisateur se trouvant de le fichier /etc/passwd, soit en rajoutant le login de l'utilisateur dans le fichier /etc/group où les noms des membres sont séparés par des virgules.

2) Le root

Le root est le seul utilisateur à avoir tous les droits sur le système. Il peut créer, modifier, effacer n'importe quel compte utilisateur ou fichier par exemple. C'est le compte de l'administrateur du système. Le root a l'UID "0" et le GID "0". ce qui permet de reconnaître le compte ayant les droits root même si son nom a été modifié dans le fichier passwd. Le répertoire personnel du root ne se trouve pas dans "/home" comme pour les autres utilisateurs. Il est conseillé d'utiliser le compte root seulement lorsque cela est indispensable. En effet, une mauvaise manipulation avec le compte root peut avoir des conséquences désastreuses pour le système.

3) Les permissions

Sous Linux, chaque fichier a des droits particuliers par rapport à :
- un utilisateur, plus exactement le propriétaire du fichier
- un groupe, en général le groupe auquel appartient le propriétaire
- tous les autres utilisateurs

Pour afficher les droits d'un fichier on utilise la commande "ls" avec l'option "-l" :

```
$ ls -l fichier
drwxrw-r-- 1 root team 10 Nov 1 15:30 fichier
```

"fichier" est un répertoire (d) et il est accessible en :
lecture/écriture/exécution (rwx) pour le propriétaire du fichier, ici le root.
lecture/écriture (rw-) pour tous les utilisateurs appartenant au groupe team.
Lecture pour tous les autres (r--).

Les droits du fichier sont tout à gauche du résultat de la commande "ls -l". La première lettre des droits indique le type du fichier (répertoire, tube, lien, etc). Les trois caractères suivants (rwx), indiquent les droits du propriétaire sur le fichier.

r --> read : autorise la lecture
w --> write : autorise l'écriture
x --> exécute : autorise l'exécution
- --> interdit un certain droit

Les 3 suivants sont les droits du groupe (rw-), et les 3 derniers (r--) sont les droits de tous les autres utilisateurs sur le fichier.

Type de fichier	Droits du propriétaire			Droits du groupe			Droits des autres utilisateurs		
d	r	w	x	r	w	-	r -	-	
répertoire	lecture	écriture	exécution	lecture	écriture	exécution interdite	lecture	écriture interdite	exécution interdite

Lorsqu'un utilisateur exécute un programme sous Linux, ce programme va se lancer avec les droits de l'utilisateur et non pas avec les droits du propriétaire du programme. Il est quand même possible de faire exécuter un programme avec les droits du groupe auquel il appartient ou avec les droits du propriétaire. Lorsqu'un programme s'exécute avec les droits du propriétaire, on dit qu'il a le bit "Suid" activé. Cela se représente par un "s" à la place du bit indiquant le droit en exécution du propriétaire :

```
$ ls -l fichier1

-rwsr-xr-x 1 root root 10 Nov 1 15:30 fichier1
```

4) Les processus

Un programme actif en mémoire sous Linux est appelé "processus". A chaque processus est attribué un PID unique qui sert de numéro d'identification pour distinguer les divers processus en mémoire. Le PID peut être utilisé pour arrêter, relancer un processus en utilisant la commande "kill" que nous verrons plus loin dans le cours. Voyons pour le moment comment interpréter le résultat d'une commande "ps -aux".

USER : indique le nom de l'utilisateur avec lequel le processus a été lancé. Le processus aura donc les mêmes droits que cet utilisateur.
PID : numéro d'identification. (PPID numéro d'identification du processus parent)
%CPU : pourcentage d'utilisation du CPU.
%MEM : pourcentage d'utilisation de la mémoire.
TTY : terminal à partir duquel a été lancé le processus.
STAT : état du processus (R=actif, S=endormi, etc...)
START : date à laquelle le processus a été lancé.
TIME : temps d'exécution cumulé
COMMAND : commande ayant lancé le processus.

Lorsqu'un processus est lancé il obtient les droits de l'utilisateur l'ayant activé. Lorsqu'un processus fils est créé, il hérite du même environnement et des mêmes fichiers ouverts que le processus l'ayant créé, c'est-à-dire le processus parent. Le premier processus lancé sur un système Linux est "init". Ce processus est le seul qui est activé par le noyau, ensuite les nouveaux processus sont créés grâce à des duplications de processus existants.

- L'état d'un processus: actif, endormi ou zombie

Etat	Description
actif	Le processus est en pleine activité.
endormi	Le processus est au repos et peut être appelé par le système pour effectué une tache.
Zombie	C'est un processus qui est mort mais dont le processus parent n'a pas été informé de sa mort.

PARTIE II : Pratique

Dans cette partie nous allons étudier l'utilisation du shell et des commandes de base sous Linux.

I) Le Shell

Le shell désigne l'interpréteur de commandes. Il permet à l'utilisateur de communiquer avec le système. Le shell est totalement indépendant des commandes externes qui se trouvent dans "/bin" par exemple. Il existe plusieurs shells sous Unix, dont voici les plus importants :

- **sh** le shell POSIX
- **csh** le C-shell
- **ksh** le Korn shell
- **bash** le shell de Linux

La commande "man" peut être utilisée pour avoir plus d'informations sur un de ses shells. Les fonctionnalités de base sont quasi identiques, ce qui change c'est surtout la syntaxe utilisée pour les commandes internes. Il est possible de connaître le type (externe ou interne) d'une commande via la commande "type":

```
$ type cd
cd is a shell builtin

$ type id
ID is /usr/bin/ID
```

Le prompt du shell peut être représenté de plusieurs manières suivant la valeur de la variable "$PS1" (se trouvant dans le fichier profile). Dans notre cours, le shell est représenté par un simple caractère "$". Mais cela peut changer, en général on aura :

```
crashfr@poste1:/bin$
```

"crashfr" indiquant le nom de l'utilisateur qui utilise le shell, "poste1" le nom de la machine où il est connecté et "bin/" le répertoire dans lequel il se trouve.

1) Les Meta caractères

Linux est sensible à la casse, il est donc important de bien respecter les majuscules et minuscules lorsque l'on tape une commande ou autres. Quelques caractères spéciaux sont indispensables à connaître."*" et "?" sont ce que l'on appelle des jokers, c'est-à-dire qu'ils permettent de remplacer des caractères que l'on ne connaît pas.
"*" remplace n'importe quelle suite de caractères.
"?" Remplace uniquement un seul caractère.

```
$ ls
temp fichier1 fichier2 f1
$ ls f*
fichier1 fichier2 f1
$ ls f?
f1
```

Il est aussi possible de définir une suite de caractères à rechercher grâce aux crochets :

```
$ ls f[149]
f1
```

Ci-dessus, on met entre crochets les caractères qui peuvent être affichés, c'est-à-dire 1, 4 et 9.

Ci-dessous on définit une plage de caractères, c'est-à-dire de 0 à 9

```
$ ls f[0-9]
f1
```

2) Les commandes de base

Voici un bref descriptif des commandes de base. Pour avoir plus d'infos sur une commande et avoir les options supplémentaires, il suffit d'utiliser la commande man.

man

La commande "man" affiche le manuel d'utilisation d'une commande, d'un fichier de configuration et d'autres choses utiles comme la table ascii par exemple.

```
$ man ascii
$ man ls
```

cd

La commande "cd" permet de se déplacer dans les différents répertoires du système.

Syntaxe : cd [répertoire]

Commande	*Description*
cd	Renvoie l'utilisateur dans son répertoire"home". "/home/crashfr" par exemple pour l'utilisateur crashfr.
cd /	Renvoie l'utilisateur à la racine du système.
cd ../	Renvoie l'utilisateur dans le répertoire parent.
cd nomrepertoire	Déplace l'utilisateur dans le répertoire nommé "nomrepertoire".

chmod

"chmod" permet de changer les droits des utilisateurs sur les fichiers.
"chmod" s'utilise de deux manières.

Voici la première :

```
$ chmod u+w nomfichier
```

Ci-dessus, on rajoute les droits en écriture au propriétaire du fichier nommé "nomfichier" se trouvant dans le répertoire courant. Le premier argument de la commande "chmod" peut être décomposé en trois parties.
Le "u" (première partie) indique le propriétaire du fichier.
u comme *user* pour indiquer le propriétaire.
g comme *group* pour indiquer le groupe.
o comme *others* pour indiquer les autres utilisateurs.
a comme *all* pour indiquer tout le monde (u, g et o).
Le "+" indique qu'il faut ajouter un droit. Pour supprimer un droit, il faut mettre un "-" à la place.

```
$ chmod g-r nomfichier
```

Enlève le droit en écriture pour tous les utilisateurs appartenant au groupe associé au fichier s'appelant "nomfichier".
La troisième partie indique les permissions à ajouter ou à supprimer.

r comme *read* pour la lecture.
w comme *write* pour l'écriture.
x comme *execute* pour l'exécution.
s pour l'activation du bit *Set User ID*.
t pour l'activation du *sticky bit*.

La deuxième méthode pour utiliser chmod est la suivante :

```
$ chmod 754 nomfichier
```

La commande ci-dessus :
- autorise la lecture/écriture/exécution pour le root. (7)
- autorise la lecture/exécution pour le groupe. (5)
- autorise la lecture pour les autres. (4)

En fait, un chiffre est associé à chaque droit, ensuite si le droit est actif on ajoute ce chiffre aux autres.
r --> 4
w --> 2
x --> 1
Imaginons que l'on désire assigner les droits r/w/x au propriétaire du fichier, r/x pour le groupe et pour les autres.
- droits propriétaire = 4+2+1
- droits groupe = 4+1
- droits autres = 4+1

ce qui donne :

```
$ chmod 755 fichier
```

cp

"cp" permet de copier un fichier ou répertoire.

```
$ cp nom1 nom2
```

Cette commande fait une copie nommée "nom2" du fichier "nom1".

```
$ cp -r reper1/ reper2
```

Celle-ci copie le répertoire "reper1" et tout son contenu à la racine (option -r) sous le nom de "reper2".

head

La commande "head", affiche les 10 premières lignes d'un fichier.

```
$ head nomfichier
```

tail

La commande "tail" affiche les 10 dernières lignes d'un fichier.

```
$ tail nomfichier
```

cat

La commande "cat" affiche le contenu d'un fichier sans interruption vers la sortie standard (STDOUT).

```
$ cat nomfichier
```

Il est aussi possible avec "cat" de créer un fichier en utilisant une redirection ">":

```
$ cat > nouveaufichier
ce qui est tapé ici se
trouvera dans le fichier
nommé nouveaufichier
^D
```

note : le "^D" représente les touches Ctrl+D.

kill

La commande kill permet de tuer (arrêter) un processus en envoyant un signal (SIGTERM en général). Ce signal est envoyé par le noyau vers une application.

```
$ kill 900
```

"900" représente le PID de l'application à tuer. Je vous rappelle que pour afficher les processus il faut utiliser la commande "ps -aux". Il est possible de spécifier le signal à envoyer. Pour afficher les différents signaux :

```
$ kill -l
```

Pour envoyer un signal particulier:

Syntaxe : kill signal PID

ln

La commande "ln" permet de créer des liens. Plus exactement des liens durs ou des liens symboliques. Le lien dur crée un fichier indépendant du fichier source mais physiquement, utilisant le même espace sur le disque. Si l'on modifie le lien dur, celui-ci modifiera le fichier source et inversement, si l'on modifie le fichier source, il modifiera le lien dur.

```
$ ln fichierexistant nomlien
```

Le lien symbolique, quant à lui, renvoie vers un répertoire ou fichier existant sur le disque. Comme un raccourci sous Windows, donc physiquement sur le disque le fichier n'est présent qu'une fois sur le disque. Le lien symbolique est beaucoup plus petit en taille que le lien en dur.

```
$ ln -s fichierexistant nomlien
```

ls

La commande "ls" liste les fichiers contenus dans un répertoire mais il permet aussi de lister les droits, propriétaires, groupes, dates de création, tailles de chacun des fichiers.

```
$ ls -lsa
```

Option "-a" affiche les fichiers cachés. Les fichiers cachés sous Linux commencent par un ".".
Option "-l" liste le contenu du répertoire en format détaillé.
Option "-s" donne la taille de chaque fichier.

mkdir

La commande "mkdir" permet de créer un répertoire.

```
$ mkdir -p /home/crashfr/newrep
$ mkdir reper1
```

La première commande crée un répertoire"newrep" en créant si besoin le répertoire parent "crashfr" s'il n'existe pas. La seconde crée simplement un répertoire nommé "reper1" dans le répertoire courant.

mv

mv déplace un fichier ou répertoire :

```
$ mv rep1/ /home/crashfr/newrep
```

Déplace "rep1" en le renommant "newrep" dans le répertoire"/home/crashfr".

more et less

"more" et "less" affichent le contenu d'un fichier en plusieurs fois, contrairement à la commande "cat", qui affiche le contenu du fichier sans effectuer d'arrêt entre chaque page affichée.

```
$ more /etc/passwd
```

affiche passwd en partant du début du fichier

```
$ more +/crashfr /etc/passwd
```

Affiche le contenu de passwd en partant du mot "crashfr".

```
$ less /etc/passwd
```

"less" est un peu plus évolué que "more". Il permet entre autres, de revenir en arrière dans la visualisation d'un fichier grâce à la touche b.

ps

ps affiche les processus se trouvant en mémoire.

```
$ ps
```
Affiche les processus lancés à partir du terminal actuel.

```
$ ps -aux
```
Affiche tous les processus du système.

pwd

La commande "pwd" affiche le chemin complet du répertoire dans lequel l'utilisateur se trouve.
```
$ pwd
```

rm

La commande "rm" permet de supprimer un fichier ou répertoire.
```
$ rm nomdufichier
```

Pour effacer tout le contenu d'un répertoire contenant des sous-répertoires, il faut utiliser la récursivité (option -r) :
```
$ rm -r nomdufichier
```

echo

echo affiche ce que l'on lui donne en argument.
```
$ echo toto
toto
```
echo est aussi très utile pour afficher la valeur des variables. Ainsi, pour afficher la variable PATH :
```
$ echo $PATH
/bin:/usr/bin:/usr/local/bin:/usr/games
```

II) Comprendre le shell

Nous n'avons pas encore parlé des trois entrées/sorties standards que l'on trouve sur le shell. Le shell utilise une entrée standard (STDIN) pour recevoir les commandes. Donc par défaut l'entrée standard est ce que l'utilisateur tape dans son shell (clavier). Il utilise aussi une sortie standard (STDOUT) pour renvoyer le résultat. Par défaut la sortie est la console ou le terminal courant, qui renvoie cette sortie sur l'écran. Et la dernière, qui est également renvoyée vers le moniteur, renvoie les erreurs (STDERR). Ce qui est bien sous Linux c'est que l'on peut rediriger ces entrées/sorties.

1) Les redirections

Il est possible de mettre le résultat d'une commande dans un fichier grâce aux redirections. Normalement le résultat est envoyé vers l'écran de l'utilisateur, en redirigeant la sortie standard (STDOUT), le résultat sera inscrit dans un fichier.

```
$ echo toto
toto
$ echo toto > test1
$ cat test1
toto
```

La redirection ">" crée un nouveau fichier à chaque exécution et écrase le fichier s'il existe déjà.
La redirection ">>" ajoute le résultat à la suite du fichier existant. S'il n'existe pas, le fichier est créé.

L'entrée standard (STDIN) peut aussi être modifiée pour exécuter des commandes provenant d'un fichier, au lieu du clavier de l'utilisateur.

```
$ More message
Subject: Test redirection
Message à envoyer via mail à crashfr
$ mail crashfr < message
```

L'opérateur "|" appelé"pipe" permet de rediriger le résultat d'une commande vers une autre commande.

```
$ cat /etc/passwd | grep crashfr
crashfr:x:1000:1000::/home/crashfr:/bin/sh
```

2) Les caractères de contrôle.

Le ";" permet de taper plusieurs commandes à la suite sur la même ligne.

```
$ echo "salut"; echo "crashfr"; id
salut
crashfr
uid=1000(crashfr) gid=1000(crashfr) groups=1000(crashfr)
```

Le "&" permet de mettre un processus en tâche de fond de manière à rendre la main à l'utilisateur.

```
$ ethereal &
[1] 529
```

Note : le PID du processus est affiché. "ethereal" s'est vu assigné le PID 529.

Le "&&" est pratiquement identique au ";". Il permet d'exécuter une suite de commandes sur la même ligne, mais contrairement au ";" il va vérifier l'exécution de la première commande avant de lancer la suivante et

ainsi de suite. Si l'exécution de la première commande a provoqué une erreur, la seconde ne sera pas exécutée.

```
$ cat ffhei && echo "toto"
cat: ffhei: No such file or directory
$ touch ffhei
$ cat ffhei && echo "toto"
toto
```

Le "||" permet d'exécuter une commande si la première a renvoyé une erreur.

```
$ ls ffhh || echo yopyop
ls: ffhh: No such file or directory
yopyop
$ touch ffhh
$ ls ffhh || echo yopyop
ffhh
```

La première ligne de commande exécute "echo yopyop" car le fichier ffhh n'existe pas. La seconde fois, "ls ffhh" est exécuté mais pas "echo yopyop" car le fichier ffhh existe.

3) Les constantes caractères :

Les constantes permettent surtout de représenter les caractères non visibles comme les tabulations, sauts de ligne etc...

\b retour arrière
\c pas de saut de ligne
\f saut de page
\n saut de ligne
\r retour chariot
\t tabulation horizontale
\v tabulation verticale
**** caractère "\"
\xnn caractère ascii de valeur hexadécimale nn

Note : pour afficher la table ascii tapez "man ascii".

```
$ echo -e "crashfr"
crashfr
$ echo -e "\x63rash\tfr"
crash fr
```

La première commande fait un simple "echo" de "crashfr", la seconde inclut des constantes caractères qui remplacent le "c" de "crash" par sa valeur ascii hexadécimale (63) et rajoute une tabulation entre "crash" et "fr" avec la constante "\t".

4) Les utilitaires pratiques

grep

grep est un outil très puissant qui permet de rechercher une chaîne de caractères ou une expression régulière dans un fichier texte. Vous pouvez utiliser "man" pour en savoir un peu plus sur les expressions régulières.

```
$ man regexp
```

Voici quelques exemples d'utilisation de grep:

Syntaxe : grep [options] exp_reg fichier

Recherche d'une chaîne de caractères:

```
$ grep crashfr /etc/passwd
crashfr:x:1000:1000::/home/crashfr:/bin/sh
```

On demande à grep de rechercher la chaîne de caractères "crashfr" dans le fichier "passwd".

Une autre façon de faire la même recherche:

```
$ cat /etc/passwd | grep crashfr
crashfr:x:1000:1000::/home/crashfr:/bin/sh
```

On utilise "cat" pour afficher le fichier passwd et on renvoie STDOUT vers la commande "grep" grâce au pipe.

Recherche d'expressions régulières :

```
$ grep '/bin/sh$' /etc/passwd
```

"grep" recherche toutes les lignes qui finissent par "/bin/sh". C'est le caractère "$" qui ancre la recherche sur la fin de la ligne. On peut ancrer la recherche en début de ligne avec le caractère "^":

```
$ grep '^crash' /etc/passwd
crashfr:x:1000:1000::/home/crashfr:/bin/sh
crashtr:x:1001:1001::/home/crashtr:/bin/sh
crash:x:1002:1002::/home/crash:/bin/bash
```

"grep" nous renvoie trois lignes commençant par "crash".
"egrep" est une variante de "grep" qui inclut quelques fonctions supplémentaires.

```
$ egrep '^crash(fr|tr)' /etc/passwd
crashfr:x:1000:1000::/home/crashfr:/bin/sh
crashtr:x:1001:1001::/home/crashfr:/bin/sh
```

"egrep" permet, entre autres, l'utilisation des parenthèses pour grouper les expressions régulières comme ci-dessus.

cut

"cut" est un outil qui va nous permettre de découper une ligne ou un fichier pour en récupérer la ou les colonnes qui nous intéressent.

Syntaxe: cut [options] [fichier]

```
$ cat /etc/passwd | grep '^crash' | cut -f 6 -d :
/home/crashfr
/home/crashtr
```

Cette commande renvoie le contenu de passwd vers un "grep" qui va à son tour renvoyer les lignes qui commencent par "crash" vers un "cut". A ce moment précis, voilà ce qui est envoyé à "cut" :

```
crashfr:x:1000:1000::/home/crashfr:/bin/sh
crashtr:x:1001:1001::/home/crashtr:/bin/sh
```

La commande "cut -f 6 -d :" va donc renvoyer uniquement le champ 6, qui représente les répertoires"home" des utilisateurs dans le fichier "passwd". Chaque champ est délimité par ":", ce delimiteur est défini après l'options -d. L'option -f indique à "cut" de renvoyer seulement le ou les champs sélectionnés.

sed

"sed" est un filtre programmable qui combine les expressions régulières et les commandes d'édition de "ed". Ce qui nous permet de remplacer une chaîne de caractères dans un fichier texte à partir du shell sans ouvrir d'éditeur de texte.

Syntaxe: sed 'commande d'édition' fichier

```
$ more test
the hackademy school
the hackademy forum
$ sed 's/ie/y/g' test
the hackademy school
the hackademy forum
```

Ci-dessus, on demande à "sed" de remplacer 'ie' par 'y' dans tout le fichier "test". On va décomposer la partie 'commande d'édition' de sed.

```
s/ie/y/g
```

Chaque sous-partie est séparée par un "/".
La première sous-partie qui contient la lettre "s" indique le type d'édition :
s --> substitution
La deuxième sous-partie indique l'expression ou caractère à rechercher. Ici on recherche donc "ie" et on désire le remplacer par la troisième sous-partie, c'est-à-dire "y".
La dernière sous-partie, le "g", lui indique que l'on veut changer toutes les occurrences de "ie" sur chaque ligne. Si le "g" n'était pas présent, "sed" ne modifierait que la première occurrence de "ie" sur chaque ligne. Dans notre cas, cela renverrait le même résultat car nous n'avons qu'une seule occurrence de "ie" sur chaque ligne. Notez que le résultat est affiché sur votre terminal et que le fichier test n'a pas été modifié. Il faut donc rediriger la sortie standard (STDOUT) vers un fichier "test2" pour enregistrer les modifications. En effet sed modifie le fichier directement à partir du flux de données qui lui est envoyé en entrée et redirige le résultat vers la sortie standard.

```
$ sed 's/ie/y/g' test > test2
$ more test2
the hackademy school
the hackademy forum
$ mv test2 test
```

awk

awk est un autre filtre semblable à "sed", encore plus puissant, faisant tout ce qui est possible avec "sed" mais en plus on peut lui demander des instructions écrites dans un langage de script proche du langage C. C'est une sorte d'interpréteur de langage C. Au niveau de son fonctionnement awk est identique aux autres filtres existants à part qu'il ne redirige pas directement le résultat vers la sortie standard : il associe une variable à chaque champs séparée par un délimiteur. Par défaut les champs sont délimités par les espaces ou les tabulations.

Syntaxe : awk [options] 'programme awk' [fichier]

```
$ ps
PID TTY TIME CMD
848 pts/3 00:00:00 bash
849 pts/3 00:00:00 ps
$ ps | awk '{ print $2" "$4 }'
TTY CMD
pts/3 bash
pts/3 ps
pts/3 bash
```

Ici, on filtre le résultat renvoyé par la commande ps de manière à n'afficher que les champs TTY et CMD. "awk" associe les variables suivantes :
$1 à la première colonne
$2 à la seconde colonne qui est délimitée par un espace par exemple
$3 à la troisième colonne etc...

Si le délimiteur de champs est autre que l'espace ou la tabulation, il est modifiable avec l'option "-F".

```
$ ls | awk -F . '{print $1" extension: "$2}'
sudoers extension:
sysctl extension: conf
syslog extension: conf
terminfo extension:
timezone extension:
updatedb extension: conf
vsftpd extension: chroot_list
vsftpd extension: conf
wgetrc extension:
```

On a séparé le nom de l'extension pour chaque fichier présent dans le répertoire courant.

sort

"sort" a pour fonction principale de trier par ordre alphabétique les lignes d'un fichier mais "sort" peut aussi fusionner deux fichiers grâce à l'option "-m".

Syntaxe : sort [options] [fichier]

```
$ echo test
crashfr
xdream
fozzy
brotha
$ sort -n test
brotha
crashfr
fozzy
xdream
$ sort -r test
xdream
fozzy
crashfr
brotha
```

Comme vous pouvez vous en rendre compte, l'option -n trie les lignes dans l'ordre alphabétique, et l'option -r dans l'ordre inverse.

tr

"tr" échange des caractères imprimables par d'autres dans tout le contenu d'un fichier.

Syntaxe: tr char1 char2

```
$ cat /etc/passwd | tr : ! > test
$ cat test
crashfr!x!1000!1000!!/home/crashfr!/bin/sh
```

Ci-dessus on remplace simplement tous les ":" présents dans le fichier passwd par "!".

wc

wc compte le nombre de bytes, mots, lignes et caractères contenus dans un fichier.
Syntaxe: wc [options] [fichier]

```
$ wc -c test
151 test
```

Option	*Description*
-c	compte le nombre d'octets.
-m	compte le nombre de caractères.
-l	compte le nombre de lignes.
-w	compte le nombre de mots.

PARTIE III : Utilisation quotidienne

I) les éditeurs de texte

1) vi

Vi est l'éditeur de texte plein écran (contrairement aux éditeurs ligne par ligne) présent sur tous les systèmes Unix par défaut donc également sur Linux. Indispensable à connaître même si son utilisation n'est pas des plus simples. "vi" fonctionne en deux modes :
- mode commande
Ce qui est tapé est interprété par "vi" comme une commande. Pour basculer en mode commande il suffit d'appuyer sur la touche "Esc".
- mode insertion
Quand on se trouve en mode insertion, tout ce qui est tapé sur le clavier est considéré comme le contenu du fichier qui est en cours d'édition. Pour passer en mode insertion, appuyer sur "i" ou "a".

Utilisation de "vi"

Pour modifier ou créer un nouveau fichier :

```
$ vi nomduchier
```

Dès que "vi" a ouvert votre fichier, je vous conseille d'activer la fonction "showmode". Pour cela, appuyez sur "Esc" (le curseur devrait se mettre sur la ligne du bas), ensuite tapez :

```
:set showmode
```

A partir de maintenant, vi vous indiquera en bas à droite, dans quel mode (insertion, commande) il se trouve. Pour rendre cette option permanente, éditez (ou créez) le fichier ".exrc" dans votre répertoire personnel (/home/user), et ajouter la ligne suivante :

```
set showmode
```

Il existe d'autres options que l'on peut activer en utilisant la commande ":set".
Pour afficher les options actives tapez ":set" sans argument.

Les principales commandes (vi doit être en mode commande):

Commande	*Description*
:w	Enregistrer le fichier.
:q	Quitter vi.
:q!	Quitter vi sans enregistrer les modifications du fichier.
x	Effacer un caractère
:numéro	Déplacer le curseur à la ligne numéro "numéro" (ex: ":10", déplace le curseur à la ligne 10)
/chaîne	Rechercher une chaîne vers l'avant (bas).
?chaîne	Rechercher une chaîne vers l'arrière (haut).
dd	Supprimer une ligne.
$	Déplacer le curseur à la fin de la ligne
^	Déplacer le curseur au début de la ligne

Il existe beaucoup de commandes que je ne détaillerai pas ici que vous pourrez trouver en faisant un "man vi".

2) emacs

"emacs" est l'éditeur le plus puissant que vous pourrez trouver sous Linux. Cet éditeur propose un nombre incroyable de modes ce qui le rend très gros mais indispensable car très complet.

Les modes de emacs	
Mode	*Description*
Fundamental	Mode par défaut
Text	Editeur de texte
Mail	Ecrire et consulter ses mails
Gnus	Lire les newsgroups
View	Visualiseur
Shell	Lance un shell
Telnet	Client telnet
Outline	Editeur de squelettes
Indented text	Indenter du texte
Nroof	Mode nroff
Tex	Editeur tex
Latex	Editeur latex
C	Editeur C
C++	Editeur C++
Java	Editeur Java
Fortran	Editeur fortran
Lisp	Editeur Lisp
Emacs lisp	Editeur de fonctions emacs
Tetris	Jeu tetris
Doctor	Mode psychanalyste
Hanoi	Résolution des tours de hanoi

Principales commandes :
Les commandes s'utilisent sous emacs avec une combinaison de touches. On utilisera la touche "Ctrl" et "Alt". "Ctrl" sera représenté par la lettre "C" et "Alt" par la lettre "M". A la place de la touche "Alt" vous pouvez utiliser la touche "Meta", "Edit" ou "Esc".

Commande	*Description*
Mx + mode	Active le mode demandé. Pour activer le mode "tetris" par exemple, il faudra appuyer sur la touche "Alt" en restant enfoncé, touche "x", et on relâche les deux touches. Ensuite on tapera le mode suivi de la touche "Entrée".
CxCf +nom	Ouvre un fichier.
CxCs	Enregistre le fichier courant.
Ck	Coupe la ligne courante.
Cy	Colle ce qui a été copié.
C_	Undo (annule une action).
Cs	Recherche de chaîne
Cx2	Divise la fenêtre en deux dans le sens horizontal.
Cx3	Divise la fenêtre en deux dans le sens vertical.
CxCb	Liste tous les buffers (chaque fichier en cours d'édition se trouve dans un buffer)
Cxk	Détruit un buffer (Fermer un fichier)
Cx1	Garde le buffer actuellement sélectionné et enlève les autres fenêtres.

II) Linux sur le net

Nous allons voir dans cette partie les outils pour le Web en mode console qui sont livrés avec toutes les distributions. Ces outils ne sont pas les plus pratiques que l'on puisse trouver à cause de leur mode d'exécution (console). Mais leur petite taille leur donne quand même un très gros avantage dans le domaine des réseaux.

1) lynx

Lynx est un navigateur en mode console qui va nous permettre de surfer sur tous les sites Web de contenu texte. Le problème avec lynx (et son équivalent plus moderne : links) c'est qu'il ne permet pas d'afficher images, animations flash, java etc...Je vous rassure, il existe d'autres navigateurs sous Linux (Mozilla, Konqueror, etc...) qui prennent en charge tous ces formats mais il vous faudra absolument un serveur X. Ce qui peut ne pas être le cas dans certaines circonstances (lorsque la carte graphique n'est pas bien reconnue par X).

Syntaxe: lynx [options] [URL]

```
$ lynx
```

Lancez lynx sans option ni URL pour voir les commandes que nous propose lynx :

h : affiche l'aide.
o : affiche le panneau de configuration.
p : imprime la page web actuelle.
g + URL : permet d'indiquer l'url du site à visiter.
m : retour à la page d'accueil du site défini dans les options.
q : quitte lynx (validez par "entrée" ou appuyez sur "n" pour annuler)
/ mot : permet de rechercher un mot dans la page courante.
backspace : la touche "backspace" affiche l'historique.
space : la touche "espace" fait défiler la page.
Touche directionnelle bas : descend dans les liens de la page.
Touche directionnelle haut : remonte dans les liens de la page.
Touche directionnelle gauche : retour à la page précédente.
Touche directionnelle droite : suivre le lien sélectionné

2) nn

"nn" est un petit client permettant de lire les messages des groupes de news. Après avoir installé "nn", vous pouvez vous connecter à un serveur en le spécifiant avec l'option "nntp-server" comme ci-dessous :

```
$ nn nntp-server=news.wanadoo.fr
```

Une fois connecté au serveur (le temps de chargement est assez long car il récupère tous les en-têtes des messages), vous devriez voir les messages du premier groupe de news, ici : a.bsu.programming.

Principales commandes : voir tableau page suivante.

Commande	*Description*
U	S'abonner, se désabonner d'un groupe.
N	Groupe suivant.
P	Groupe précédent.
Y	Lister tous les groupes.
G	Aller dans un groupe précis.
R	Répondre à un message.
:post C	Poster un nouveau message.
.	Sélectionner un message.
Z	Lire le message sélectionné.
?	Afficher l'aide.

3) telnet, ftp, ssh

"telnet" est une application cliente qui, comme vous l'aurez deviné, est utilisée pour se connecter à un serveur telnet (port 23). Le protocole Telnet qui fait partie de l'ensemble "TCP/IP" permet à un client de se connecter sur un serveur pour avoir un invite de commande distant (remote shell). Cela permet au client d'avoir accès à son compte comme s'il se trouvait derrière le clavier du serveur distant. Le serveur telnet est de moins en moins utilisé de nos jours car son gros défaut est qu'il laisse transiter en clair le login et password de tous les utilisateurs. Mais le client telnet a d'autres utilisations qui le rend tout de même utile. On peut très bien se connecter et envoyer un mail via un serveur smtp ou utiliser telnet comme client irc etc...

Syntaxe : telnet nom_du_serveur port

"ftp" est un client pour les serveurs FTP (port 21) mais aussi un protocole faisant lui aussi partie de l'ensemble "TCP/IP". Ce protocole est utilisé pour le transfert de fichiers. Cette commande (comme telnet) est disponible sur tous les systèmes par défaut.

Syntaxe : ftp [options] ipserveur

Voici les commandes FTP les plus importantes :

Commande	*Description*
ls	Lister le contenu du répertoire courant.
cd	Changer de répertoire.
pwd	Afficher le chemin du répertoire courant.
mkdir répertoire	Créer un répertoire.
get fichier	Télécharger un fichier vers notre ordinateur.
mget	Prendre tous les fichiers du répertoire distant.
put fichier	Uploader un fichier vers le serveur.
mput	Envoyer tous les fichiers du répertoire local vers le serveur.
delete fichier	Effacer un fichier sur le serveur.
help	Afficher l'aide.
close	Fermer la connexion.
quit	Quitter le client ftp.

Tableau page 26

"ssh" est un client pour serveur SSH (port 22). SSH est apparu pour rendre plus sécurisée la connexion à distance comme telnet. En effet ssh crypte la communication, donc le login et le password des utilisateurs. Ce qui rend impossible le vol de comptes utilisateurs par une personne mal intentionnée qui utiliserait ce que l'on appelle le "sniffing", mais cela ne rend pas pour autant le protocole ssh absolument sans failles. SSH reste tout de même beaucoup plus sécurisé que telnet grâce à son système de cryptage à double clés (publique/privée), qui assure aussi l'authentification du serveur (et optionnellement du client).

Syntaxe : ssh [options] login@ipserveur

Principales options :

Option	*Description*
-p	Spécifie le port.
-2	Indique que le protocole utilisé est ssh2.
-V	Affiche la version.
-h	Affiche l'aide.
+C	Active la compression.
-C	Désactive la compression.
-i fichier	Indique le fichier où se trouve notre clé publique.

4) Mutt

"mutt" va vous permettre de consulter, gérer et envoyer des mails. Le fichier de configuration de "mutt" se trouve dans votre répertoire personnel sous le nom de ".muttrc".

Les principales commandes :

Commande	*Description*
entrée	lire le message sélectionné
d	effacer un message
u	récupérer le message effacé
m	envoyer un mail
r	répondre à un message
q	quitter mutt

III) Introduction à l'installation et à la compilation d'un programme

Il existe plusieurs méthodes pour installer un logiciel sous Linux. Soit on installe notre programme à partir d'un paquetage, soit à partir des sources en effectuant une compilation. L'installation via un paquetage est plus simple qu'à partir des sources.

1) Installation de paquetage .deb

L'installation de paquetages ressemble un peu à une installation sous Windows. On double-clique et c'est installé. Suivant les distributions de Linux, l'outil et la méthode permettant l'installation de paquetages changent. Sous Debian, il existe un outil très utile et puissant permettant l'installation, la désinstallation, la mise à jour des paquetages à partir d'un cdrom ou directement à partir du web. Cet outil se nomme "apt". Pour changer la source d'installation (par défaut, le cdrom) il faut modifier le fichier /etc/apt/sources.list. Pour définir comme sources le web à la place du cdrom, il suffit de décommenter les lignes correspondant

aux serveurs http et commenter la ligne correspondant à votre cdrom.

Principales commandes :

Commande	*Description*
apt-get install paquetage	Installer un paquetage
apt-get remove paquetage	Enlever un paquetage
apt-cache search paquetage	Rechercher un paquetage

Ici on recherche l'application nommée "hping". "apt" nous renvoie tous les résultats ayant un rapport avec "hping". Ensuite on installe "hping2", on exécute "hping2" et on enlève le paquetage. Pour lister tous les paquetages installés, utilisez la commande:

```
$ dpkg -l
```

La commande "dpkg" permet aussi l'installation de paquetages. En fait, "apt" est un outil, qui est utilise "dpkg".

Syntaxe : dpkg option paquetage

```
$ dpkg -i /home/crashfr/hping.deb
```

Installe le paquetage "hping.deb".

Principales options :

Option	Description
-i	Installer un paquetage.
-r	Enlever un paquetage.
-l	Rechercher un paquetage
-L	Afficher plus d'informations sur un paquetage.

2) Installation des paquetages RPM

Si vous n'utilisez pas uneDebian mais plutôt une Mandrake, Redhat, Suse ou toute autre RedHat like, le format des paquetages ne sera plus ".deb" mais ".rpm". Pour installer les ".rpm" la commande "rpm" est nécessaire à connaître sauf si vous utilisez un Manager graphique comme par exemple "rpmdrake" sous Mandrake, qui ne nécessite aucune commande pour installer/désinstaller un RPM.

Nous allons tout de même apprendre à utiliser "rpm" dans le cas où vous n'avez pas de serveur X fonctionnant correctement.
Pour notre exemple, nous allons installer le paquetage de "nmap" que vous pourrez trouver sur:

http://www.insecure.org/nmap/nmap_download.html

Syntaxe : rpm options paquetage

Principales options:

Option	*Description*
-v	mode verbose (affiche les informations sur le déroulement des opérations)
-i	installer un nouveau paquetage
-U	mettre à jour ou installer un paquetage en enlevant les versions déjà installées
-e	désinstaller un paquetage

L'outil "rpm" permet de récupérer et d'installer des paquetages à partir d'un serveur http ou sur un support physique (cdrom, lecteur de disquette, etc...). Pour "nmap" on pourrait donc taper :

```
$ rpm -vi http://download.insecure.org/nmap/dist/nmap-3.00-1.i386.rpm
Retrieving http://download.insecure.org/nmap/dist/nmap-3.00-1.i386.rpm
Preparing packages for installation...
nmap-3.00-1
```

La commande ci-dessus va télécharger le paquetage à l'URL indiquée et l'installer sur votre système. Vous pouvez aussi lui indiquer un paquetage que vous avez déjà téléchargé sur votre machine :

```
$ ls
nmap-3.00-1.i386.rpm
$ rpm -vi nmap-3.000-1.i386.rpm
Preparing packages for installation...
nmap-3.00-1
```

Pour désinstaller un paquetage, rien de plus simple. Il vous suffit de lui indiquer l'option "-e" et de lui donner le nom de l'application à désinstaller, comme ceci :

```
$ rpm -e nmap
```

3) Installation à partir des sources

Maintenant que vous savez installer un paquetage, nous allons nous attaquer à la compilation de fichiers. La compilation a pour but de transformer un fichier texte en un exécutable (un programme binaire). Le fichier texte contient une suite d'instructions écrites en un certain langage (ex: le langage C) suivant le compilateur utilisé. La plupart du temps quand vous téléchargerez les sources d'un programme, elles seront compressées dans un fichier ayant l'extension ".tar.gz". Pour décompresser le fichier il suffit d'utiliser l'utilitaire de décompression "tar". Dans notre exemple, nous allons installer "xbill" qui est un petit jeu sous Linux que vous pourrez trouver sur **http://www.xbill.org .**

```
$ ls
xbill-2.1.tar.gz
$ tar -zvxf xbill-2.1.tar.gz
xbill-2.1/
xbill-2.1/bitmaps/
xbill-2.1/bitmaps/apple.xbm
xbill-2.1/bitmaps/bsd.xbm
xbill-2.1/bitmaps/bucket.xbm
xbill-2.1/bitmaps/hand_down.xbm
xbill-2.1/bitmaps/hand_down_mask.xbm
xbill-2.1/bitmaps/hand_up.xbm
xbill-2.1/bitmaps/hand_up_mask.xbm
xbill-2.1/bitmaps/hurd.xbm
```

```
xbill-2.1/bitmaps/linux.xbm
xbill-2.1/bitmaps/next.xbm
xbill-2.1/bitmaps/os2.xbm
xbill-2.1/bitmaps/palm.xbm
xbill-2.1/bitmaps/redhat.xbm
xbill-2.1/bitmaps/sgi.xbm
xbill-2.1/bitmaps/sun.xbm
xbill-2.1/.cvsignore
xbill-2.1/Bill.c
xbill-2.1/Bill.h
xbill-2.1/Bucket.c
xbill-2.1/Bucket.h
xbill-2.1/Cable.c
xbill-2.1/Cable.h
xbill-2.1/ChangeLog
xbill-2.1/Computer.c
xbill-2.1/Computer.h
xbill-2.1/Game.c
xbill-2.1/Game.h
xbill-2.1/Horde.c
xbill-2.1/Horde.h
xbill-2.1/INSTALL
xbill-2.1/Makefile.in
xbill-2.1/Network.c
xbill-2.1/Network.h
xbill-2.1/OS.c
xbill-2.1/OS.h
xbill-2.1/README
xbill-2.1/README.Ports
xbill-2.1/Scorelist.c
xbill-2.1/Scorelist.h
xbill-2.1/Spark.c
xbill-2.1/Spark.h
xbill-2.1/UI.c
xbill-2.1/UI.h
xbill-2.1/acconfig.h
xbill-2.1/config.h.in
xbill-2.1/config.in
xbill-2.1/configure
xbill-2.1/configure.in
xbill-2.1/gtk.c
xbill-2.1/gtk.h
xbill-2.1/install-sh
xbill-2.1/mkinstalldirs
xbill-2.1/scores
xbill-2.1/strings.h
xbill-2.1/types.h
xbill-2.1/util.c
xbill-2.1/util.h
xbill-2.1/x11-athena.c
xbill-2.1/x11-athena.h
xbill-2.1/x11-motif.c
xbill-2.1/x11-motif.h
xbill-2.1/x11.c
```

```
xbill-2.1/x11.h
xbill-2.1/xbill.6
xbill-2.1/xbill.gif
xbill-2.1/xbill.spec
xbill-2.1/pixmaps/
xbill-2.1/pixmaps/about.xpm
xbill-2.1/pixmaps/apple.xpm
xbill-2.1/pixmaps/billA_0.xpm
xbill-2.1/pixmaps/billA_1.xpm
xbill-2.1/pixmaps/billA_10.xpm
xbill-2.1/pixmaps/billA_11.xpm
xbill-2.1/pixmaps/billA_12.xpm
xbill-2.1/pixmaps/billA_2.xpm
xbill-2.1/pixmaps/billA_3.xpm
xbill-2.1/pixmaps/billA_4.xpm
xbill-2.1/pixmaps/billA_5.xpm
xbill-2.1/pixmaps/billA_6.xpm
xbill-2.1/pixmaps/billA_7.xpm
xbill-2.1/pixmaps/billA_8.xpm
xbill-2.1/pixmaps/billA_9.xpm
xbill-2.1/pixmaps/billD_0.xpm
xbill-2.1/pixmaps/billD_1.xpm
(...)
xbill-2.1/pixmaps/wingdows.xpm
```

Ci-dessus, la décompression du fichier "xbill-2.1.tar.gz" effectuée avec "tar".

```
$ cd xbill-2.1
$ ls
Bill.c Bill.h Bucket.c Bucket.h Cable.c Cable.h
ChangeLog Computer.c Computer.h Game.c Game.h Horde.c
Horde.h INSTALL Makefile.in Network.c Network.h OS.c
OS.h README README.Ports Scorelist.c Scorelist.h Spark.c
Spark.h UI.c UI.h acconfig.h bitmaps config.h.in
config.in configure configure.in gtk.c gtk.h install-sh
mkinstalldirs pixmaps scores strings.h tmp types.h
util.c util.h x11-athena.c x11-athena.h x11-motif.c x11-motif.h
x11.c x11.h xbill.6 xbill.gif xbill.spec
```

Nous nous trouvons maintenant dans le répertoire "xbill-2.1/" où se trouvent tous les fichiers permettant la compilation de notre jeu. Si vous faites attention au listing, vous remarquerez la présence d'un fichier "configure". Le fichier "configure" est un exécutable généré par "autoconf", qui va vérifier si toutes les dépendances nécessaires sont présentes sur votre système pour pouvoir installer le programme correctement. En effet pour qu'un programme puisse être compilé, il peut avoir besoin d'une certaine librairie par exemple, cette librairie fera donc partie des dépendances nécessaires à la compilation. Nous allons maintenant afficher les options de "configure", ce qui va nous permettre de modifier le répertoire d'installation, le répertoire où se trouvent les librairies etc... Il peut être utile de forcer le répertoire d'installation lorsque l'on n'est pas root sur le système.

```
$ ./configure --help
Usage: configure [options] [host]
Options: [defaults in brackets after descriptions]
Configuration:
--cache-file=FILE cache test results in FILE
```

```
--help print this message
--no-create do not create output files
--quiet, --silent do not print `checking...' messages
--version print the version of autoconf that created configure
Directory and file names:
--prefix=PREFIX install architecture-independent files in PREFIX
[/usr/local]
--exec-prefix=EPREFIX install architecture-dependent files in EPREFIX
[same as prefix]
--bindir=DIR user executables in DIR [EPREFIX/bin]
--sbindir=DIR system admin executables in DIR [EPREFIX/sbin]
--libexecdir=DIR program executables in DIR [EPREFIX/libexec]
--datadir=DIR read-only architecture-independent data in DIR
[PREFIX/share]
--sysconfdir=DIR read-only single-machine data in DIR [PREFIX/etc]
--sharedstatedir=DIR modifiable architecture-independent data in DIR
[PREFIX/com]
--localstatedir=DIR modifiable single-machine data in DIR [PREFIX/var]
--libdir=DIR object code libraries in DIR [EPREFIX/lib]
--includedir=DIR C header files in DIR [PREFIX/include]
--oldincludedir=DIR C header files for non-gcc in DIR [/usr/include]
--infodir=DIR info documentation in DIR [PREFIX/info]
--mandir=DIR man documentation in DIR [PREFIX/man]
--srcdir=DIR find the sources in DIR [configure dir or ..]
--program-prefix=PREFIX prepend PREFIX to installed program names
--program-suffix=SUFFIX append SUFFIX to installed program names
--program-transform-name=PROGRAM
run sed PROGRAM on installed program names
Host type:
--build=BUILD configure for building on BUILD [BUILD=HOST]
--host=HOST configure for HOST [guessed]
--target=TARGET configure for TARGET [TARGET=HOST]
Features and packages:
--disable-FEATURE do not include FEATURE (same as --enable-FEATURE=no)
--enable-FEATURE[=ARG] include FEATURE [ARG=yes]
--with-PACKAGE[=ARG] use PACKAGE [ARG=yes]
--without-PACKAGE do not use PACKAGE (same as --with-PACKAGE=no)
--x-includes=DIR X include files are in DIR
--x-libraries=DIR X library files are in DIR
--enable and --with options recognized:
--with-x use the X Window System
--enable-motif build with the Motif widget set
--enable-athena build with the Athena widget set
--enable-gtk build with the GTK widget set
```

Ces options sont spécifiques à xbill-2.1, si vous installez un autre programme les options ne seront pas identiques mais certaines sont génériques. L'option que l'on va modifier dans notre exemple, sera le répertoire d'installation. Par défaut, le programme sera installé dans un répertoire où vous n'avez pas le droit d'écrire en temps que simple utilisateur (/usr/bin, /bin, etc). L'option pour modifier le répertoire d'installation est "prefix" :

```
$ ./configure --prefix=$HOME/bin
```

Ici j'ai spécifié à "configure" que le répertoire d'installation est $HOME/bin, donc le répertoire /bin se trouvant dans mon répertoire "home" (/home/crashfr/bin). Si le "configure" se déroule sans erreur, plusieurs fichiers sont créés, dont un config.h et un config.cache contenant les paramètres détectés, ainsi qu'un fichier Makefile s'il doit être généré par "autoconf" à l'aide de "Makefile.in". Le fichier Makefile est utilisé pour compiler notre programme, il nous évite de spécifier à "gcc", notre compilateur, les différentes options et fichiers qu'il doit compiler. Pour utiliser un Makefile, il vous faudra l'outil "make".

```
$ make -v
GNU Make 3.80
Copyright (C) 2002 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.
There is NO warranty; not even for MERCHANTABILITY or FITNESS FOR A
PARTICULAR PURPOSE.
```

La commande ci-dessus affiche la version de "make" si la commande est installée sur votre machine. Regardons ce que contient notre fichier Makefile :

```
$ more Makefile
# Generated automatically from Makefile.in by configure.
CC = gcc
CFLAGS = -g -O2
CPPFLAGS =
LDFLAGS =
LIBS = -L/usr/X11R6/lib -L/usr/lib -L/usr/X11R6/lib -lgtk -lgdk -rdynamic -lgmodule -lglib -
ldl -lXi -lXext -lX11 -lm -lSM -lICE
INSTALL=/usr/bin/install -c
INSTALL_DATA=${INSTALL} -m 644
MKINSTALLDIRS = $(SHELL) $(top_srcdir)/mkinstalldirs

X_CFLAGS= -I/usr/X11R6/include
GTK_CFLAGS=-I/usr/include/gtk-1.2    -I/usr/include/glib-1.2    -I/usr/lib/glib/include    -
I/usr/X11R6/include

prefix=/home/crashfr/bin
exec_prefix=${prefix}
bindir=${exec_prefix}/bin
mandir=${prefix}/man
datadir=${prefix}/share
localstatedir=${prefix}/var
top_srcdir = .
srcdir = .

ALL_CFLAGS = $(CFLAGS) $(CPPFLAGS) -I$(srcdir)
ALL_LDFLAGS = $(CFLAGS) $(LDFLAGS)
IMAGE_DEFINES=-DIMAGES=\"${datadir}/xbill\"
SCORE_DEFINES=-DSCOREFILE=\"${localstatedir}/xbill/scores\"

PROG = xbill

OBJS = Bill.o Bucket.o Cable.o Computer.o Game.o Horde.o Network.o \
OS.o Scorelist.o Spark.o UI.o util.o gtk.o

XPMS = pixmaps/about.xpm pixmaps/apple.xpm pixmaps/billA_0.xpm \
pixmaps/billA_1.xpm pixmaps/billA_10.xpm pixmaps/billA_11.xpm \
```

```
pixmaps/billA_12.xpm pixmaps/billA_2.xpm pixmaps/billA_3.xpm \
pixmaps/billA_4.xpm pixmaps/billA_5.xpm pixmaps/billA_6.xpm \
pixmaps/billA_7.xpm pixmaps/billA_8.xpm pixmaps/billA_9.xpm \
pixmaps/billD_0.xpm pixmaps/billD_1.xpm pixmaps/billD_2.xpm \
pixmaps/billD_3.xpm pixmaps/billD_4.xpm pixmaps/billL_0.xpm \
pixmaps/billL_1.xpm pixmaps/billL_2.xpm pixmaps/billR_0.xpm \
pixmaps/billR_1.xpm pixmaps/billR_2.xpm pixmaps/bsd.xpm \
pixmaps/bsdcpu.xpm pixmaps/bucket.xpm pixmaps/hurd.xpm \
pixmaps/icon.xpm pixmaps/linux.xpm pixmaps/logo.xpm \
pixmaps/maccpu.xpm pixmaps/next.xpm pixmaps/nextcpu.xpm \
pixmaps/os2.xpm pixmaps/os2cpu.xpm \
pixmaps/palm.xpm pixmaps/palmcpu.xpm pixmaps/redhat.xpm \
pixmaps/sgi.xpm pixmaps/sgicpu.xpm pixmaps/spark_0.xpm \
pixmaps/spark_1.xpm pixmaps/sun.xpm pixmaps/suncpu.xpm \
pixmaps/toaster.xpm pixmaps/wingdows.xpm

XBMS = bitmaps/apple.xbm bitmaps/bsd.xbm bitmaps/bucket.xbm \
bitmaps/hand_down.xbm bitmaps/hand_down_mask.xbm bitmaps/hand_up.xbm \
bitmaps/hand_up_mask.xbm bitmaps/hurd.xbm bitmaps/linux.xbm \
bitmaps/next.xbm bitmaps/os2.xbm bitmaps/palm.xbm bitmaps/redhat.xbm \
bitmaps/sgi.xbm bitmaps/sun.xbm

MANDIR = man6
MAN = xbill.6

all: ${PROG}

xbill: ${OBJS}
$(CC) $(ALL_LDFLAGS) -o $@ ${OBJS} ${LIBS}

.c.o:
$(CC) $(ALL_CFLAGS) -c $< -o $@

Scorelist.o: Scorelist.c
$(CC) $(ALL_CFLAGS) $(SCORE_DEFINES) -c $< -o $@

x11-motif.o: x11-motif.c
$(CC) $(ALL_CFLAGS) $(X_CFLAGS) -c $< -o $@

x11-athena.o: x11-athena.c
$(CC) $(ALL_CFLAGS) $(X_CFLAGS) -c $< -o $@

x11.o: x11.c
$(CC) $(ALL_CFLAGS) $(X_CFLAGS) $(IMAGE_DEFINES) -c $< -o $@

gtk.o: gtk.c
$(CC) $(ALL_CFLAGS) $(GTK_CFLAGS) $(IMAGE_DEFINES) -c $< -o $@

install:
$(MKINSTALLDIRS) ${DESTDIR}${bindir}
$(INSTALL) ${PROG} ${DESTDIR}${bindir}/${PROG}
$(MKINSTALLDIRS) ${DESTDIR}${mandir}/${MANDIR}
$(INSTALL_DATA) ${srcdir}/${MAN} ${DESTDIR}${mandir}/${MANDIR}
$(MKINSTALLDIRS) ${DESTDIR}${localstatedir}/xbill
```

```
$(INSTALL_DATA) -m 0666 ${srcdir}/scores ${DESTDIR}${localstatedir}/xbill
$(MKINSTALLDIRS) ${DESTDIR}${datadir}/xbill
$(MKINSTALLDIRS) ${DESTDIR}${datadir}/xbill/pixmaps
$(MKINSTALLDIRS) ${DESTDIR}${datadir}/xbill/bitmaps
for i in ${XPMS}; do \
${INSTALL_DATA} ${srcdir}/$$i ${DESTDIR}${datadir}/xbill/pixmaps ; \
done
for i in ${XBMS}; do \
${INSTALL_DATA} ${srcdir}/$$i ${DESTDIR}${datadir}/xbill/bitmaps ; \
done

distclean::
rm -f config.cache config.h config.log config.status Makefile

distclean clean::
rm -f ${PROG} *.o
rm -f *core core*
```

La première chose que l'on peut voir, c'est que l'on retrouve bien la valeur du répertoire d'installation que l'on a forcé :
prefix=/home/crashfr/bin
Le fichier Makefile définit beaucoup de variables dont :
CC : Correspond au compilateur utilisé. Ici on utilise gcc.
CFLAGS : Options du compilateur.
prefix : Répertoire d'installation.
PROG : Le nom du programme.
OBJS : Contient les noms des fichiers objets (noms des fichiers une fois compilés).

En suite le fichier Makefile est décomposé en plusieurs sections délimitées par un mot, suivi de ":". Dans ce Makefile nous avons 11 parties :

```
all

xbill
.c.o
Scorelist.o
x11-motif.o
x11-athena.o
x11.o
gtk.o
install
distclean
distclean clean
```

Chacune de ces parties va permettre d'effectuer une opération que l'on pourra donner en argument à la commande "make" s'il l'on se trouve dans un répertoire contenant le fichier Makefile. En général, pour compiler et installer un soft via un Makefile on tapera les commandes suivantes :

```
$ make
```
Pour compiler le programme.

```
$ make install
```
Pour l'installer dans un répertoire prédéfini.

Quand vous tapez "make install", "make" exécute ce qui se trouve dans le Makefile après le mot "install:". Si l'on s'intéresse un peu à la partie "distclean" du fichier Makefile, on voit de suite que la commande "make distclean" va effacer tous les fichiers qui ont été créés par "autoconf" via l'utilisation du fichier "configure".

Pour installer "xbill" on pourrait donc procéder comme ceci :

```
$ make
```

Si la commande "make" se termine correctement, votre programme sera compilé et placé dans le répertoire courant. Pour vérifier que "xbill" fonctionne tapez :

```
$ ./xbill
```

Maintenant il ne reste plus qu'à le mettre dans un répertoire du système où tous nos programmes seront installés, c'est-à-dire $HOME/bin dans notre cas.

```
$ make install
```

Une fois l'installation terminée, il ne vous reste plus qu'à taper le nom du soft dans votre console.

```
$ xbill
```

Si jamais le shell ne trouve pas la commande, c'est que le programme ne se trouve pas dans un des répertoires définis grâce à la variable d'environnement $PATH.

IV) Les fichiers de configuration

La plupart des fichiers de configuration sur Linux se trouvent dans le répertoire /etc . Tous ces fichiers peuvent être modifiés avec un simple éditeur de texte.

1) Configuration de inetd

Inetd est un super serveur qui est lancé au démarrage de Linux et lançant d'autres serveurs (daemons). Ce super serveur écoute sur différents ports de la machine et dès qu'une connexion est établie, "inetd" charge le serveur nécessaire suivant le fichier de configuration /etc/inetd.conf . Nous allons voir de suite à quoi ressemble ce fichier :

```
$ cat /etc/inetd.conf
# /etc/inetd.conf: see inetd(8) for further information.
#
# Internet server configuration database
#
# <service_name> <sock_type> <proto> <flags> <user> <server_path> <args>
#
#:INTERNAL: Internal services
#echo stream tcp nowait root internal
#echo dgram udp wait root internal
#chargen stream tcp nowait root internal
#chargen dgram udp wait root internal
discard stream tcp nowait root internal
discard dgram udp wait root internal
daytime stream tcp nowait root internal
#daytime dgram udp wait root internal
time stream tcp nowait root internal
```

```
#time dgram udp wait root internal
#:STANDARD: These are standard services.

#:BSD: Shell, login, exec and talk are BSD protocols.

#:MAIL: Mail, news and uucp services.
smtp stream tcp nowait mail /usr/sbin/exim exim -bs
```

Dans inetd.conf on peut voir plusieurs lignes représentant chacune un service. Prenons pour exemple la ligne :

```
daytime stream tcp nowait root internal
```

Le premier paramètre, "daytime" indique à "inetd" le nom du serveur et son port. A chaque nom est attribué un port, la liste des noms est consultable dans le fichier /etc/services . Si vous faites un "cat" sur ce fichier vous remarquerez que le port 13 est réservé pour "daytime".
Le second paramètre, "stream" est le type de socket utilisé. Pour le protocol TCP on utilisera stream, pour le protocole UDP ça sera dgram.
Le troisième paramètre c'est le protocole utilisé. TCP ou UDP.
Le quatrième paramètre, "nowait" indique que le serveur sera en multithread, "wait" qu'il sera en monothread.
Le sixième paramètre, "root" indique avec quels droits doit être exécuté le serveur.
Le dernier paramètre, "internal" représente le chemin où se trouve le serveur suivi de ses options. "internal" signifie que le serveur est implémenté dans inetd. Vous remarquerez dans le fichier inetd d'exemple, que le serveur smtp "exim" se trouve dans "/usr/sbin/".

Tous les serveurs n'utilisent pas forcément inetd pour fonctionner, vous pouvez vous en rendre compte si vous commentez toutes les lignes du fichier inetd.conf. Faites un scan de vos ports avec "nmap". Si vous avez encore des ports ouverts c'est que certains serveurs ne passent pas par inetd. L'utilité de inetd sera étudiée plus en détails lorsque l'on abordera "xinetd".

2) Une approche de l'administration de votre système

1. init

"init" est le seul processus lancé par le noyau, ensuite "init" se charge de lancer les autres processus nécessaires au fonctionnement de notre système. Le fichier de configuration de init est "inittab". On peut, entre autres, y définir le niveau d'exécution par défaut. Il existe dans le répertoire "etc/" pour la Debian plusieurs répertoires nommés "rcx.d" où "x" est compris entre 0 et 7. A chaque répertoire est associé un niveau d'exécution suivant la valeur de "x". Tous ces répertoires sont aussi présents dans les "Red Hat Like" mais dans le répertoire "/etc/rc.d". Le tableau qui suit est spécifique à la Debian, sous Red Hat les niveaux sont un peu différents.

Répertoire	Niveau	Description
rc0.d/	0	Arrêt
rc1.d/	1	Mono-utilisateurs
rc2.d/	2	Multi-Utilisateurs
rc3.d/	3	Multi-Utilisateurs
rc4.d/	4	Multi-Utilisateurs
rc5.d/	5	Multi-Utilisateurs
rc6.d/	6	Redémarrage

Vous trouverez aussi un répertoire "init.d/" contenant un script pour chaque processus pouvant être exécuté au démarrage. Par exemple, si l'on désire lancer "exim", il suffit de taper :

```
$ /etc/init.d/exim start
```

Pour l'arrêter :

```
$ /etc/init.d/exim stop
```

Si l'on regarde ce qui se trouve dans un des répertoires "rcx.d", on trouvera des liens symboliques vers les scripts se trouvant dans le répertoire "inet.d/". Si le lien commence par la lettre "S" c'est qu'il va lancer un processus, si c'est une lettre "K", c'est que ce lien va arrêter un processus. Les deux chiffres qui suivent n'ont pas de signification particulière.

```
$ pwd
/etc/rc2.d
$ ln -s ../init.d/exim S99exim
```

Cette commande va créer un lien symbolique dans le répertoire "/etc/rc2" qui pointera vers le script "../init.d/exim" pour lancer le serveur "exim" au démarrage de Linux, en mode muti-utilisateurs.

2.cron

La commande "cron" combinée avec le fichier/commande "crontab", vous permettra de planifier des tâches périodiques. "cron" est lancé une fois au démarrage de Linux, il va ensuite vérifier toutes les minutes s'il y a une tâche à effectuer dans le fichier "/etc/crontab". Voici un exemple de fichier crontab :

```
$ cat /etc/crontab
# Unlike any other crontab you don't have to run the `crontab'
# command to install the new version when you edit this file.
SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin
# m h dom mon dow user command
25 6 * * * root test -e /usr/sbin/anacron || run-parts --report /etc/cron.daily
47 6 * * 7 root test -e /usr/sbin/anacron || run-parts --report /etc/cron.weekly
52 6 1 * * root test -e /usr/sbin/anacron || run-parts --report /etc/cron.monthly
#
```

Le fichier crontab contient une variable SHELL et PATH dont vous devriez connaître l'utilité. Intéressons-nous à une tâche planifiée :

```
25 6 * * * root test -e /usr/sbin/anacron || run-parts --report /etc/cron.daily
```

Pour définir la plage horaire il y a cinq variables.

Variable	Description
m	minute de 0 à 59
h	heure de 0 à 23
dom	jour dans mois de 01 à 31
mon	mois dans l'année de 01 à 12
mow	jour dans la semaine de 01 à 07

Après la plage horaire il y a le nom de l'utilisateur utilisé, suivi de la tâche (commande) à exécuter. Dans notre exemple :

```
m=25
h=6
dom=*
mon=*
mow=*
```

Donc la commande "test -e /usr/sbin/anacron || run-parts --report /etc/cron.daily" sera exécutée avec les droits root tous les jours à 6h25. Le root peut éditer directement le fichier "crontab" pour planifier des tâches système. Il est aussi possibles pour les utilisateurs de créer leur fichier crontab personnel, en utilisant la commande "crontab" dont voici les principales commandes :

```
$ crontab -l
```

Liste les tâches planifiées

```
$ crontab -e
```

Permet de créer un fichier crontab

```
$ crontab -r
```

Enlève le fichier crontab actuel

3. e2fsck

"e2fsck" permet de faire un scan du disque dur pour tenter de réparer les inodes.

```
$ e2fsck -y /dev/hda1
```

Fait un scandisk de la partition hda1 en activant la réparation automatique avec l'option "-y".

V) Utilisation du mode graphique

Linux ne fonctionne pas qu'en mode console, heureusement d'ailleurs... Pour avoir une interface graphique gérant le système de fenêtres X-Window, il vous faudra un serveur X, un gestionnaire de fenêtres et un gestionnaire de sessions graphiques.

1) Le serveur X

Dans notre cas, on choisira XFree86 comme serveur X qui est disponible sur http://www.xfree86.org et installé par défaut sur la plupart des distributions. L'installation de XFree86 sera abordée dans le niveau 2 du cours Linux. C'est lui qui va s'occuper de prendre en charge la carte graphique, la résolution , le nombre de couleurs, la souris, le clavier pour permettre à n'importe quel programme/utilisateur de se connecter dessus en tant que client.

2) Le gestionnaire de fenêtres

C'est le gestionnaire de fenêtres qui permet d'utiliser plusieurs fenêtres contenant chacune une application graphique ou un shell. Il s'occupe aussi de tous les éléments graphiques de votre environnement de travail (barre des tâches, menu démarrer, raccourcis bureau, etc..). Mandrake utilise par défaut, le gestionnaire KDE mais il en existe d'autres, comme Gnome et Enlightenment par exemple.

3) Le gestionnaire de sessions xdm et l'export display

"xdm" s'occupe de gérer les sessions sur le serveur X ainsi qu'un ensemble de "displays".
L'"export display" est une technique permettant d'afficher sur notre serveur X des applications installées sur un client distant ne disposant pas de serveur X. Pour notre exemple, nous allons exporter un xterm (termi-

nal graphique) à partir d'un serveur ssh, vers notre serveur X. Pour commencer, il faut autoriser les connexions distantes via le port 6000 à notre serveur X, pour cela il faut modifier le contenu du fichier "/etc/X11/xdm/Xservers" dont voici le contenu :

```
# $Xorg: Xserv.ws.cpp,v 1.3 2000/08/17 19:54:17 cpqbld Exp $
#
# Xservers file, workstation prototype
#
# This file should contain an entry to start the server on the
# local display; if you have more than one display (not screen),
# you can add entries to the list (one per line). If you also
# have some X terminals connected which do not support XDMCP,
# you can add them here as well. Each X terminal line should
# look like:
# XTerminalName:0 foreign
#
# When adding X servers, please keep the following things in mind:
# - For each X server other than :0, the display number must be on the command
# line.
# - This is the place to add the -bpp option to the command line.
# - It is wise to add the vtXX option to each local X server, since this is
# the only sure way of knowing in advance which X server attaches to which
# console.
# - See the Xserver(1) manpage for a description of X server options.
#
# Examples for multiple local X displays:
# :0 local /usr/X11R6/bin/X :0 vt9 -bpp 16
# :1 local /usr/X11R6/bin/X :1 vt10 -bpp 8
:0 local /usr/X11R6/bin/X vt7 -dpi 100 -nolisten tcp
```

Il suffit de commenter la dernière ligne et décommenter :
":0 local /usr/X11R6/bin/X :0 vt9 -bpp 16"

Pour que xdm puisse prendre en compte le changement en compte il faut le redémarrer :

```
$ killall -HUP xdm
```

Voilà, maintenant les connexions au serveur X via le port 6000 sont possibles mais il faut quand tout de même autoriser les connexions provenant de n'importe quel hôte :

```
$ xhost +
access control disabled, clients can connect from any host
```

Il ne reste plus qu'à établir une connexion au serveur ssh et lui demander de nous renvoyer une application (xterm) vers notre machine. Ici l'adresse 192.168.2.6 représente notre serveur X et l'IP 192.168.2.112 le client X (serveur ssh) :

```
$ ssh 192.168.2.112
$ xterm -display 192.168.2.6:0
```

Vous pouvez maintenant lancer n'importe quelle application se trouvant sur le serveur ssh à partir du xterm apparu sur votre bureau. Les applications seront affichées sur votre machine mais vous utiliserez la puissance de la machine distante (serveur ssh). Voici une capture avec un "gkrellm" local (bleu) et un distant (rouge). gkrellm est une application qui permet d'avoir l'état d'un machine (utilisation cpu, disque dur, réseau etc...), un petit outil très pratique pour les administrateurs.

PARTIE IV : Les scripts Shell

Un script, contrairement à un exécutable, n'a pas besoin d'être compilé. Un script contiendra tout simplement une suite d'instructions qui seront directement interprétées par notre shell dans le cas d'un script shell. Cette partie vous donnera une première approche à la programmation.

1) Comment les définir ?

Tout script sous Linux commence par le caractère "#" (caractère de commentaires) suivi d'un point d'exclamation. Ensuite on trouve le chemin complet jusqu'à l'interpréteur de commandes. Exemple :

```
#!/bin/sh
```

Ici le script sera exécuté par l'interpréteur (shell) sh se trouvant dans le répertoire "/bin". L'interpréteur va lire notre script ligne par ligne. On aurait très bien pu créer un script Perl en indiquant l'interpréteur Perl, en général, "/usr/bin/perl". Pour lancer un script, il faut autoriser son exécution avec "chmod" ou en indiquant le chemin complet de shell suivi du nom du script à lancer.

```
$ cat test
#!/bin/sh
echo The Hackademy
$ /bin/sh test
The Hackademy
$ chmod +x test
$./test
The Hackademy
```

Les script shell facilitent l'administration et la maintenance d'un système Unix.

2) Les variables

Il est très courant de définir des variables dans les scripts shell. Une variable est définie par un nom composé de lettres, chiffres et une valeur. Une variable peut avoir deux portées différentes :
- locale
Une variable locale n'est disponible que pour le shell l'ayant définie. La commande "set" permet d'afficher toutes les variables définies pour le shell courant.
- globale
Une variable globale est disponible pour les processus fils lancés par le shell l'ayant définie. Une fois définie comme globale, la variable devient une variable d'environnement. La commande "env" affiche les variables d'environnement.

```
$ cours=linux
$ echo $cours
linux
$ set
cours=linux
$ export cours
$ env
cours=linux
```

Ci-dessus, on crée la variable nommée cours ayant comme valeur la chaîne de caractères "Linux". Notez que pour afficher la valeur d'une variable grâce à "echo", il faut précéder le nom de la variable par un caractère "$". La commande "set" est ensuite exécutée pour montrer que la variable "cours" fait bien partie des variables locales. "export" permet de transformer une variable locale en variable globale, c'est pour cela qu'elle apparaît ensuite dans les variables d'environnement. Pour démontrer la portée des variables nous

allons utiliser deux petits fichiers, test1 et test2.

```
$ cat test1
echo "variable locale: $var1"
echo "variable globale: $var2"
$ cat test2
#!/bin/sh
var1=the
var2=hackademy
export var2
./test1
$ chmod +x test*
$ ./test2
variable locale:
variable globale: hackademy
$ rm test*
```

Le script shell test2 va donc déclarer deux variables, une locale et une globale. On voit très bien que lorsque test1 est exécuté, la variable $var1 n'a aucune valeur, seule la variable globale est affichée. L'utilisation des parenthèses permet, entre autres, de mettre le résultat d'une commande en tant que valeur d'une variable :

```
$ echo $(ls)
uid=1000(crashfr) gid=1000(crashfr) groups=1000(crashfr)
Mais on peut aussi définir une variable ayant comme résultat l'exécution d'une commande grâce
au caractère "`".
$ a = `id`
$ echo $a
uid=1000(crashfr) gid=1000(crashfr) groups=1000(crashfr)
```

3) La concaténation

La concaténation permet d'attacher (coller) deux valeurs de variables grâce aux accolades :

```
$ a=crash; b=fr
$ echo ${a}${b}
crashfr
```

4) Les arguments

Il est possible de passer des arguments à un script shell. Pour récupérer un argument, il faut utiliser le caractère "$" suivi d'un chiffre indiquant la position de l'argument. "1" indique le premier argument, "2" le second, etc...

```
$ cat test
#!/bin/sh
echo $2
$ chmod +x test
$ ./test coucou
$ ./test coucou coucou
coucou
```

"echo $2" affiche le second argument envoyé au script "test".

$*	Affiche tous les arguments.
$#	Affiche le nombre d'arguments.

5) L'instruction read

L'instruction "read" récupère une ligne tapée par l'utilisateur.

```
$ cat test
echo "Entrez un mot :\c"
read mot
echo "Vous avez tapé : $mot"
$ ./test
Entrez un mot :
toto
Vous avez tapé : toto
```

6) Les itérations

La boucle "for" :

La boucle "for" permet d'exécuter une suite d'instructions un certain nombre de fois.
Syntaxe : for variable [in valeurs] do instructions done

Exemple 1

```
$ cat test
#!/bin/sh
for var
do
     echo "Var=$var"
done
$ chmod +x test
$ ./test yopyop
Var = yopyop
```

Exemple 2

```
$ cat test
#!/bin/sh
for var in toto yopyop
do
     echo "Var=$var"
done
$ chmod +x test
$ ./test
Var=toto
Var=yopyop
```

La boucle "while" :

La boucle "while" exécute une suite d'instructions tant qu'une condition est valide.
Syntaxe : while condition do instructions done

Exemple 1 :

```
$ cat test
#!/bin/sh
echo "Tapez votre pass : "
read pass
while [ "$pass" != "yopyop" ]
do
echo "Ce n'est pas le bon password !!"
```

```
echo "Entrez à nouveau votre pass : "
read pass
done
echo "Password valide !"
$ ./test
Tapez votre pass :
default
Ce n'est pas le bon password !!
Entrez à nouveau votre pass :
yopyop
Password valide !
```

Pour la condition de la boucle "while" on peut utiliser les comparaisons.

Comparaison		*Description*
!=	ne	différent
=	eq	égal à
	lt	plus petit que
	gt	plus grand que
	le	plus petit ou égal
	ge	plus grand ou égal

Exemple 2 :

```
$ cat test
#!/bin/sh
i=1
while [ $i -lt 200 ]
do
     echo $i
     i=$[i+1]
done
```

Les sauts inconditionnels :

La commande "break" arrête l'exécution d'une suite d'instructions d'une boucle et l'exécution reprend à la fin de la boucle (après "done").
La commande "continue" arrête l'exécution d'une suite d'instructions d'une boucle et l'exécution reprend au début de la boucle.

L'instruction de contrôle "if" :

L'instruction "if" permet d'effectuer un test sur une variable par exemple et de proposer différentes alternatives d'exécution.
1ère syntaxe : if condition then instructions fi
2e syntaxe : if condition then instructions else instructions fi

Exemple :

```
if [ $i -eq 198 ]
then
     echo "la condition if est ici vraie"
fi
```

Choix multiples "case" :
"case" permet d'exécuter des instructions suivant plusieurs cas différents.
Syntaxe : case expr in modèle) instructions;; [modèle) instructions;;] esac

Exemple :

```
$ echo test
#!/bin/sh
case $1 in
10)
echo "Vous avez envoyé la valeur 10 en argument"
;;
20)
echo "Vous avez envoyé la valeur 20 en argument"
;;
*)
echo "Vous avez envoyé une valeur différente de 10 et 20"
;;
esac
```

7) Exercices

Voici quelques exercices pour récapituler tout ce que l'on vient de voir sur le shell..

Exercice 1 :
En partant de "ifconfig", créer une commande renvoyant l'adresse IP de votre carte réseau suivie de son adresse de Broadcast (Bcast).

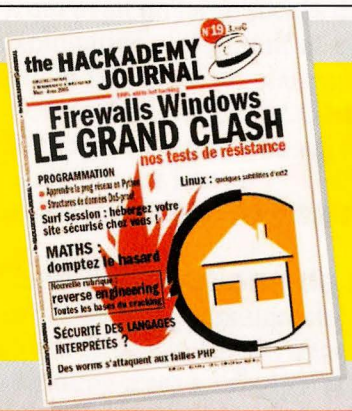
Exercice 2 :
Afficher le login, le nom et le répertoire home du root en partant du fichier passwd.

Exercice 3 :
Afficher les 10 premières lettres de l'alphabet en majuscules, via une boucle while et en utilisant les valeurs ascii hexadécimales.

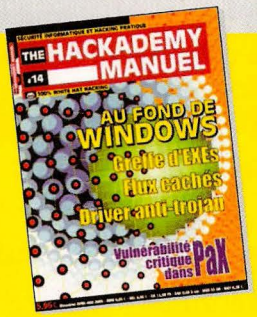
par CrashFr de The Hackademy School

Hackademy Mag mode d'emploi

A partir du 10 septembre, THJ et le Manuel se transforment en Hackademy Mag


the HACKADEMY JOURNAL
Firewalls Windows
LE GRAND CLASH
nos tests de résistance
PROGRAMMATION
Linux : quelques substituts d'ext2
Surf Session : hébergez votre site sécurisé chez vous !
MATHS : domptez le hasard
reverse engineering
Toutes les bases du cracking
SÉCURITÉ DES LANGAGES INTERPRÉTÉS ?
Des worms s'attaquent aux failles PHP

+



=

HACKADEMY #1 MAGAZINE

- 84 pages
- Format magazine
- Sortie du numéro 1, le 10 septembre 2005
- Bimestriel
- Prix de vente en kiosque : 5,95 euros (4,50 euros si commandé par internet et expédié dans les 48 heures)
- Sommaire type : 70 pages de pure technique + 14 pages de grands reportages, enquêtes, débats
- Rédacteur en chef : dvrasp
- Rédaction : Hackademy Team
- Objectif : roxer sa mère

Déjà abonnés, vous êtes privilégiés

Vous êtes abonnés au journal seul :
vous receverez le Mag de 84 pages sans formalité ni surcoût !

Vous êtes abonnés au Manuel seul :
vous receverez le Mag de 84 pages sans formalité ni surcoût !

Vous êtes abonnés au journal et au Manuel :
vous receverez le Mag de 84 pages plus 6 hors séries par an sans formalité ni surcoût !

Vous souhaitez mettre fin à votre abonnement actuel :
Nous vous remboursons au prorata sur simple demande de votre part !

Futurs abonnés, vous avez tout à gagner

Si vous souscrivez un abonnement au Mag avant le 25 août, chaque numéro vous revient à 4,50 euros :
Soit une économie de près de 9 euros, et la certitude de ne rater aucun numéro !

Promos et nouveaux packages !!

Abonnement 1 an : 27 euros • **Basic**

Abonnement 2 ans : 49 euros • **Fidelity**

Abonnement 1 an + les cours newbie et newbie⁺ de the Hackademy School :
99 euros • **Init package**

Abonnement 2 ans + les cours newbie et newbie⁺ de the Hackademy School + + le vintage t-shirt intrusion.exe :
119 euros • **I luv U package**

20 % de réduction suplémentaire pour tout produit commandé en ligne jusqu'au 25 août 2005 !!!!

Abonnez-vous !!

HACKADEMY #1 MAGAZINE	**Basic :** Abonnement 1 an	27 euros ☐
HACKADEMY #1 MAGAZINE	**Fidelity :** Abonnement 2 ans	49 euros ☐
HACKADEMY #1 MAGAZINE	**Init package :** Abonnement 1 an + les cours newbie et newbie⁺ de the Hackademy School	99 euros ☐
HACKADEMY #1 MAGAZINE	**I luv U package :** Abonnement 2 ans + les cours newbie et newbie⁺ de the Hackademy School + le vintage t-shirt *intrusion.exe*	119 euros ☐

Je calcule le montant de ma commande : ☐

PAIEMENT
☐ par chèque à l'ordre de DMP
☐ par Carte Bleue ☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐
Expire en ☐☐☐☐

20 % de réduction suplémentaire pour tout produit commandé en ligne jusqu'au 25 août 2005 !!!!
wwww.thehackademy.net

Nom :..... Prénom :.....
Adresse :.....
Ville :..... Pays :.....
E-mail :.....

Signature

Envoyez votre bulletin accompagné de votre règlement à l'ordre de D.M.P., The Hackademy 26 bis rue Jeanne d'Arc 94160 Saint Mandé

FACILE ET RAPIDE : abonnement possible par téléphone avec votre CB au 01 53 66 95 28

Que vont devenir THJ et le Manuel ?
Découvrez le projet secret de the Hackademy

N°20 3.90€

THE LAST ONE

the HACKADEMY JOURNAL

BIMESTRIEL PRATIQUE
D'INFORMATION ET D'INVESTIGATION
JUIN - JUILLET 2005

COLLECTOR

100% white hat hacking

Relevez LES DEFIS DU HACKING !

Tous les bons plans pour réussir les challenges online

- Failles web
- Astuces crypto
- Hacking UNIX
- Les meilleurs liens

"Inexistence du piratage"

DES PAIEMENTS EN LIGNE PAR CB

Qui peut y croire ? p.4

Un DoS fulgurant menace les PC Windows p.18

DOM 3,8 E - BEL 4,40 E - CH 7Fs - CAN 6,50 Scan - MAR 40 DH

THE HACKADEMY JOURNAL - THE HACKADEMY JOURNAL - THE HACKADEMY JOURNAL - THE HACKADEMY JOURNAL - THE HACKADEMY JOURNAL

En vente en kiosque

JOIN US !

NEWS

COURS

COMMUNAUTÉ

www.thehackademy.net

FORUM

CHAT

BLOGS

L 11383 - 3 - F: 6,00 € - RD